TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatik XIX

Empowering End-Users to Collaboratively Structure Knowledge-Intensive Processes

Matheus Hauder

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 17.09.2015 bei der Technischen Universität München

eingereicht und durch die Fakultät für Informatik am 08.02.2016 angenommen.

Zusammenfassung

Wissensarbeit entwickelt sich zur vorherrschenden Arbeitsform und spielt in den strategisch wichtigsten Prozessen in Organisationen eine zentrale Rolle. Zu den bedeutendsten Treibern dieser Entwicklung zählt die Automatisierung von Routinearbeit durch die stetig zunehmende Nutzung von Informationstechnik und steigender Anpassungsdruck durch technische Innovationen mit denen bestehende Produkte durch digitale Dienstleistungen ersetzt werden. Trotz dieser zunehmenden Bedeutung von wissensintensiven Prozessen (KiPs) bieten betriebliche Informationssysteme keine ausreichende Unterstützung, da existierende Workflow-Management-Systeme unstrukturierte und nicht vollständig vordefinierte Prozesse nur unzureichend unterstützen.

Aufgrund der unvorhersehbaren und emergenten Eigenschaften von Wissensarbeit ist es unverzichtbar für zukünftige betriebliche Informationssysteme, KiPs zur Laufzeit zu definieren. Als Antwort auf dieses Problem ist es das Ziel dieser Arbeit, die kollaborative Strukturierung von KiPs durch Endanwender zu ermöglichen. Zu diesem Zweck müssen Endanwender ohne Spezialkenntnisse dazu befähigt werden, Arbeitspläne zu pflegen. Mit diesen Arbeitsplänen wird die Grundlage für die Generalisierung von wiederverwendbaren Arbeitsvorlagen geschaffen, mit denen implizites Wissen dokumentiert werden kann.

Im ersten Schritt wurden generische Anforderungen für die Unterstützung von KiPs aus der Literatur und praktischen Anwendungsszenarien gesammelt. Auf Basis dieser Anforderungen entstand ein Framework für die Unterstützung von KiPs, welches aus drei Komponenten besteht: einem Lebenszyklus der die Evolution von KiPs unterstützt, einer Menge von leichtgewichtigen Strukturierungskonzepten und generischen Funktionalitäten die auf Prinzipien erfolgreicher Online Communities basieren. Das Framework wurde prototypisch umgesetzt und mit Hilfe von kontrollierten Experimenten evaluiert.

Im Rahmen der Vorstudie mit 7 Teilnehmern und der Hauptstudie mit 145 Teilnehmern wurde die Lösung in zwei unterschiedlichen Anwendungsszenarien produktiv eingesetzt. In der Vorstudie haben die Teilnehmer einen Arbeitsplan für einen vollkommen unstrukturierten KiP gemeinsam definiert. In der Hauptstudie wurde eine vordefinierte Arbeitsvorlage eingesetzt, um einen vorhandenen KiP zu unterstützen. In beiden Studien wurde die Einsatzfähigkeit der Lösung zur Unterstützung von KiPs durch Fragebögen bestätigt.

Die Ergebnisse dieser Arbeit haben zwei Implikationen für betriebliche Informationssysteme. Erstens können vorhandene Fallbearbeitungssysteme einen höheren Grad an Flexibilität ermöglichen ohne dabei die Struktur zu benachteiligen. Dadurch werden zahlreiche neue Einsatzmöglichkeiten denkbar, z. B. integrierte Versorgung, Produktentwicklung, oder hochgradig kreative und kollaborative Wissensarbeit im Allgemeinen. Zweitens können spezialisierte Softwarelösungen mit den gewonnenen Erkenntnissen erweitert werden, um prozedurales Wissen in Arbeitsvorlagen zu dokumentieren.

Abstract

Knowledge work is becoming the leading type of work and is involved in the most important processes in organizations. Main drivers for this development are the ever-increasing use of information technology that leads to an automation of routine work and growing strain of technical innovations that extrude existing products through digital services. Despite this growing relevance business information systems still lack appropriate support for knowledge-intensive processes (KiPs), since existing workflow management solutions provide no means to deal with unpredictable situations.

Due to the unpredictable and emergent characteristic of knowledge work, it is indispensable for future business information systems to define KiPs on the fly. In response to this issue the objective of this thesis is to facilitate the collaborative structuring of KiPs through end-users. For this purpose end-users without computer science background have to be empowered to maintain work plans. These work plans provide the foundation for the generalization of reusable work templates that capture implicit knowledge of end-users.

In the first step generic requirements for the support of KiPs are gathered from literature and practical application scenarios. Based on these requirements a framework for the software support of KiPs is developed that consists of three elements: a lifecycle supporting the evolution of KiPs, a lightweight set of structuring concepts, and generic features that are based on design principles from successful online communities. This framework is implemented in a software solution and evaluated through controlled experiments.

Within the preliminary study with 7 participants and the main study with 145 participants the solution has been used productively in two different application scenarios at our university. In the preliminary study the participants collaboratively defined a work plan for an entirely unstructured KiP. In the main study a predefined work template has been successfully used to support an existing KiP. In both studies an online questionnaire with the participants confirmed the ability of the solution to support KiPs.

The results of this thesis have two implications for business information systems. First, existing case management (CM) systems can leverage more flexibility without deteriorating the structure. This provides manifold new application possibilities for CM in scenarios that are much more unpredictable, e.g., integrated care, product development, highly creative and collaborative knowledge work in general. Second, special-purpose applications can be extended with our approach to capture procedural knowledge in work templates.

Acknowledgment

This research originated from my work as research assistant at the chair for Software Engineering for Business Information Systems (sebis) at the Technical University Munich. During this time advisors, colleagues, family and friends encouraged me on the long way to this success. Throughout this thesis the pronoun *we* is used to express my gratitude to everybody, who supported me during the past four years.

First and foremost, I would like to thank my advisor Prof. Dr. Florian Matthes for the opportunity to work on this extremely interesting research topic under the best possible conditions. His confidence in my work has been an incredible source of motivation to make this reality. I appreciate his intellectual guidance, helpful suggestions, and constant support throughout the last four years. I further want to express my sincere gratitude to Prof. Dr. Manfred Reichert for being my second advisor and for the valuable discussions in Ulm.

The sebis chair provided an excellent environment for the realization of my research through fruitful discussions with colleagues. I would like to thank Dr. Sascha Roth for the frequent and productive discussions on research ideas. Felix Michel has been a constant companion of my research as student assistant, master's degree candidate, as well as colleague and contributed with valuable ideas to my research. My thanks go to Pouya Aleatrati Khosroshahi, Adrian Hernandez-Mendez, Jörg Landthaler, Manoj Mahabaleshwar, Dr. Ivan Monahov, Dr. Christian Neubert, Thomas Reschenhofer, Alexander Schneider, Klym Shumaiev, Dr. Christopher Schulz, Dr. Alexander Steinhoff, Bernhard Waltl, and Marin Zec for the constructive cooperation in teaching and researching.

I would like to thank the students that wrote their thesis under my guidance or supported my research as student assistant. Thanks to Natascha Abrek, Michael Bigontina, Stefan Bleibinhaus, Jenny Cheng, Max Fiedler, Manuel Gerstner, Mario Guma, Annemarie Heuschild, Nikolaus Katinszky, Maurice Laboureur, Stefan Laner, Dominik Münch, Stephan Münter, Duc Nguyen, Simon Pigat, Daniel Richter, Daniel Sel, and Alexej Utz. It was a pleasure to work with you on exciting student projects.

A special thank goes to Yolanda Gil who supervised my master's thesis in Los Angeles and inspired me to pursue a PhD. It was a pleasure to continue the collaboration within this thesis. Several case studies were conducted in organizations and many industry partners provided valuable feedback during interviews. I want to thank all industry partners for spending their time and providing valuable practical insights for my research.

Most importantly, I want to express my gratitude to my partner Laura Kraszewski for her unconditional love and support. Thanks to my parents Janina and Christian Hauder, and my sister Dr. Johanna Munzert for supporting me during my whole life. Without this support and encouragement from my family this work would not have been possible.

Garching b. München, June 7, 2016

[illegible]

Matheus Hauder

Contents

List of Figures

List of Tables

The productivity of knowledge workers will not be the only competitive factor in the world economy. It is, however, likely to become the decisive factor, at least for most industries in the developed countries.

Peter Drucker

CHAPTER 1

Motivation

Fierce market competition and digital transformation of manufacturing industries to service providers leads to an automation and replacement of less educated workers in organizations [BM11]. At the same time the ability to constantly adapt to changing market requirements and the generation of innovations are crucial for the sustainable success. This development leads to work environments that require highly trained experts that can perform many tasks autonomously. These experts are known as knowledge workers and their processes have a tremendous impact on the success. Despite this importance existing information systems lack appropriate support for knowledge-intensive processes (KiPs), since workflow management solutions do not allow for a comprehensive support of unstructured or semi-structured processes like KiPs[1]. Davenport describes this evolution of work as follows: *"I've come to the conclusion that the most important processes for organizations today involve knowledge work. In the past, these haven't really been the focus of most organizations - improving administrative and operational processes has been easier - but they must be in future"* [Da05].

These administrative and operational processes are based on the assumption that they are characterized by routine work that consists of repeated tasks. This kind of structured work is specified by modeling experts with a prescribed execution flow of the entire process. The current maturity of these approaches has led to the application of process management in new unstructured knowledge-intensive scenarios, such as software engineering, healthcare, science and financial services. These processes are becoming increasingly important and knowledge workers represent almost half of the overall workforce of an organization [Wh09]. Due to the lower level of predictability compared to routine processes, software support for KiPs needs to balance between structured elements for repetitive elements of a process and unstructured elements to allow for creativity which is indispensable for the unpredictable nature of knowledge work.

[1] Note that there exist adaptive process management systems like ADEPT [DR09] or AristaFlow BPM Suite [Kr14] which allow for dynamic workflow changes at runtime to cope with unforeseen situations

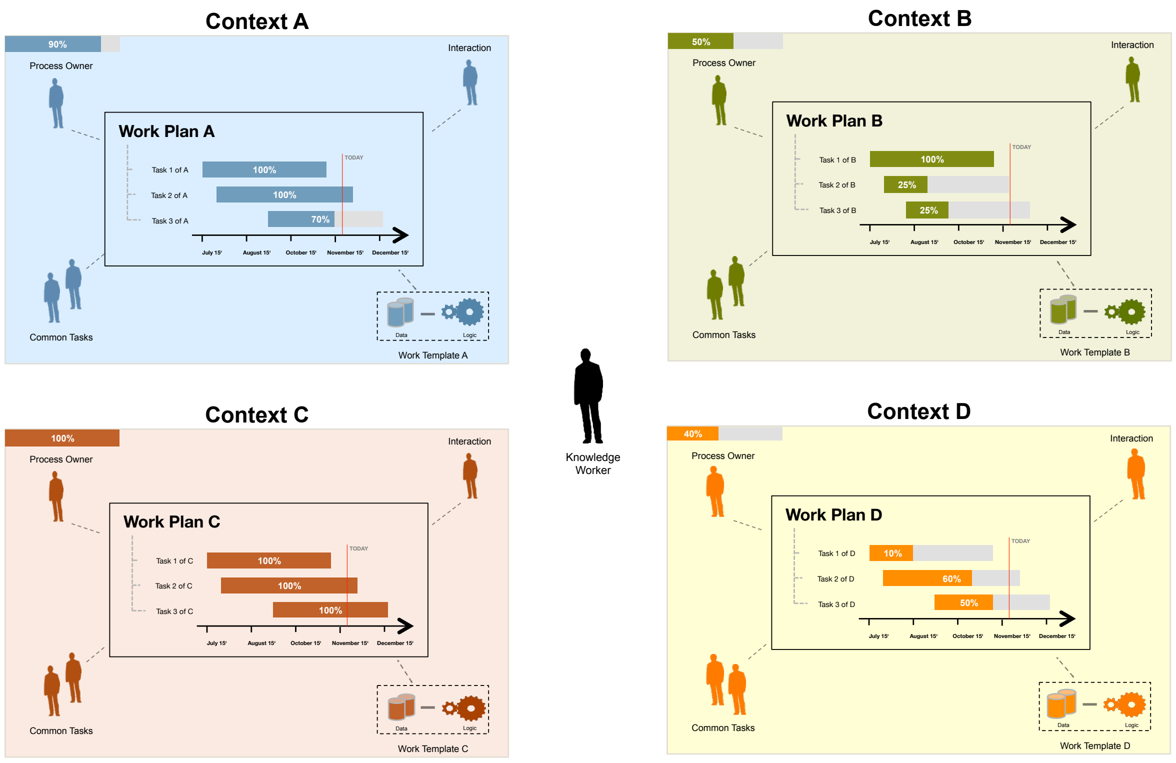


Figure 1.1.: Knowledge workers participate in KiPs in various different contexts [Mu12]

Further major differences of KiPs compared to routine processes identified in literature are the high degree of collaboration and self-organization in knowledge work [DCMR13]. KiPs emerge bottom-up through individual contributions of end-users during the process enactment, which makes the active involvement of end-users determining for the overall success of an approach supporting knowledge workers. For this purpose research on successful online communities provides valuable design principles to encourage contributions through motivation, encourage commitment, start communities and deal with newcomers [Kr12a]. Developing a sustainable online community around KiPs that constantly contributes to the definition of the process provides an interesting opportunity to realize the potential of collective intelligence.

Requirements for KiPs as well as their characteristics were recently described on a very generic level in [DCMR13, Ha14a, MKR13]. In their works the authors identified a tight integration between data elements and actions that influence each other. This requires a data-centric process perspective that needs to capture the structure, interaction and behavior of data elements. Dependencies between data elements and actions need to be described by rules and constraints to define execution semantics. Figure 1.1 illustrates these elements in the various contexts of the knowledge workers. All these elements dynamically change in relation to the actual context and environment. For all those reasons further research is needed to solve the problem of developing a software solution to support these challenging requirements for KiPs.

1.1. Problem Description

Existing solutions for workflow management are not suitable to support KiPs, since they are too rigid and provide no solutions to deal with unpredictable circumstances [ASW03]. The unpredictable nature of KiPs induces a large amount of exceptions and traditional workflow management models would become too complex to manage and maintain [SM95]. In 2001, case management has been initially promoted as a new paradigm to support the required flexibility for knowledge workers during the process [AB01]. In contrast to workflow management which is focused on what *should* be done, case management is focused on what *can* be done to achieve a certain goal [RRA03]. Despite these initial efforts software support for processes in knowledge work is far from being mastered [DCMR13]. A detailed analysis on existing case management efforts will be described in Section 2.3.

In 2009, the Object Management Group (OMG) issued a request for proposal (RFP) to create a standard modeling notation for case management[2]. Main purpose of this request was the development of a counterpart for the Business Process Model and Notation (BPMN) that supports a data-centric approach which is based on business artifacts [MHV13]. The result for this request was published in 2014 as the Case Management Model and Notation (CMMN) [Ob14]. Main difference compared to BPMN is that CMMN builds on a declarative instead of an imperative process model [MHV13]. Declarative process models specify what should be done without specifying how it should be done to facilitate the required flexibility [PCR06].

Recent literature emphasized the challenge of *knowledge worker empowerment*, which is indispensable since KiPs cannot be entirely predefined by modeling experts but require on the fly adaption by end-users during the process execution [DCMR13, HPM14, MKR13]. According to the literature review on research challenges conducted by Hauder et al. in [HPM14], the empowerment of knowledge workers is the most frequently addressed challenge in this research area. End-users with limited computer science background are usually not familiar with process modeling languages like CMMN. Another reason is that all modeling constructs of the complete CMMN specification might not be necessary in practice [MLVDP14]. Similarly, Zur Muehlen et al. [ZMR08] proposed several subsets of the BPMN language after its initial release that only contain constructs that are frequently used in practical settings.

In organizations wiki are increasingly used as shared knowledge repositories that can be used for the collaborative gathering of information. In this research, the author views these wikis as promising tools for the collaborative and self-organizing structuring of KiPs by end-users. Although wikis can be dynamically adapted to new information structures, there are limitations of existing solutions and general research challenges for this problem that are described in the following. Lightweight declarative process constructs for the definition of logical dependencies between tasks with their association to data are not supported in current wiki implementations [HPM14]. Regarding the data model it has to be possible that end-users can adapt the metamodel during the process enactment [DCMR13]. Support for the KiP lifecycle that is described by Mundbrod et al. [MKR13] is important to enable the collaborative documentation, adaptation and instantiation of templates for KiPs.

[2] `http://www.omg.org/cgi-bin/doc?bmi/09-09-23`, last accessed on: 2015-03-03

1.2. Research Design

Based on the previously described problem description, we deduce a set of research questions that guide the solution development for the science goal of software support for the collaborative structuring of processes in knowledge work. Subsequently, the applied research method is described to investigate the main research hypothesis of this thesis:

> ***Research hypothesis:*** End-users without knowledge about an existing process modeling notation can be empowered to collaboratively structure processes for knowledge work through a software solution.

In line with Gläser et al. [GL10] our research questions add new knowledge to the existing body of knowledge. The results of the research questions mainly contribute to the fields business process management (BPM), information systems (IS), intelligent user interface design (IUI), as well as computer supported cooperative work (CSCW). The research questions to investigate the main research hypotheses are described in the following:

> ***Research question 1:*** What are requirements for the software support of knowledge-intensive processes?

To answer this question, existing literature on knowledge-intensive processes and case management is analyzed. In addition, real world case studies are conducted in organizations to retrieve a thorough set of requirements for software support of KiPs. In this first step the requirements are described on a generic level without taking into account specific requirements that might only be relevant for a subset of application scenarios. A precise understanding of requirements serves as basis for the development of the theoretical framework and the software solution.

> ***Research question 2:*** What are dimensions in which knowledge-intensive processes vary from each other and how do they influence the requirements?

Based on the previously derived generic requirements for KiPs from literature, dimensions that distinguish application scenarios from each other are investigated based on three practical scenarios that are presented in this thesis. These scenarios can be considered as KiPs that exist in many organizations. In the first step the relevance of the generic requirements for the application scenarios is analyzed, i.e., to identify which requirements are more or less relevant for the given scenario.

> ***Research question 3:*** What are design principles from successful online communities and how can they be incorporated in the user interface?

The structure of KiPs emerges bottom-up through contributions of individual knowledge workers during the process. To ensure as many contributions as possible knowledge workers have to be engaged through intrinsic motivation as well as incentives in the structuring of the processes. Design principles from successful online communities provide the theoretical basis for the design of the user interface to achieve this engagement [Kr12a]. We applied design principles that are in particular relevant for the early stages of an online community, i.e., starting communities, encouraging contributions through motivation, encouraging commitment and dealing with newcomers.

> ***Research question 4:*** What is a subset of the Case Management Model and Notation with sufficient expressiveness that can be used for the structuring of knowledge-intensive processes?

Existing process modeling notations (e.g., CMMN) might overwhelm non-expert users that have no background in computer science and are not familiar with the language. Marin et al. described in [MLVDP14] that subsets of the CMMN language that are less complex for end-users are necessary, who do not need to understand and use all constructs of the complete specification in practice. For this purpose we describe a subset of lightweight structuring elements as metaphors for the CMMN language. End-users with no experience in business process modeling can use this subset to collaboratively structure KiPs. According to the literature review conducted by Hauder et al. in [HPM14], the empowerment of knowledge workers is one of the key research challenges to enable software support for KiPs.

> ***Research question 5:*** How can the lifecycle for knowledge-intensive processes be supported through a software solution?

Similar to the lifecycle of traditional business processes in BPM, processes in knowledge work also have a lifecycle that needs to be supported. This lifecycle for KiP consists of four phases for the orientation, template design, collaboration at runtime and records evaluation [MKR13]. In the framework design of thesis these four phases are integrated in the approach to enable a holistic support of KiPs. We will describe how every phase of the lifecycle is covered through the presented software solution in this thesis.

> ***Research question 6:*** How can the upcoming Case Management Model and Notation be integrated to support knowledge-intensive processes?

To answer this question, we investigate to which extend the CMMN language can be used to leverage KiPs. Before this specification is introduced, its foundation which is based on business artifacts is described briefly [MHV13]. We compare the previously identified requirements for KiPs with the capabilities of this emerging standard. Since this specification is in a very early stage implementations and experiences in practice are missing currently. Furthermore, this specification only covers the graphical modeling notation and neglects the runtime environment that needs to be provided for a holistic software support of KiPs. We will present an implementation of this specification based on the required subset for the structuring of KiPs and demonstrate its usage within three case studies.

> ***Research question 7:*** How can the structuring of knowledge-intensive processes be supported on mobile devices?

The increasing usage of mobile devices in organizations is likely to be relevant for the support of KiPs as well. Based on this research question a novel interface that is tailored to the specific requirements of mobile devices is developed. The small screen size requires adaptations of the user interface that go beyond a simple responsive design. Some features are less relevant on mobile devices and this has to be considered appropriately. Otherwise end-users might be overwhelmed with features that are unnecessary and this could impair the usability. Another challenge related to this research question is the technical architecture that is required. The mobile client needs to be integrated with the server of the software solution that is developed in this thesis through standardized interfaces.

Several methods to perform research have been introduced and applied in the domain of *information systems (IS)*. In this thesis we carefully follow the design science research methodology presented by Peffers et al. in [Pe07]. Figure 1.2 illustrates the consecutive steps that are pursued to solve the aforementioned research questions. We cover the entire design science research cycle in this PhD thesis. Every step of the research methodology is described in one section, except for the design & development step that is described in two sections since it covers all of the contributions made.

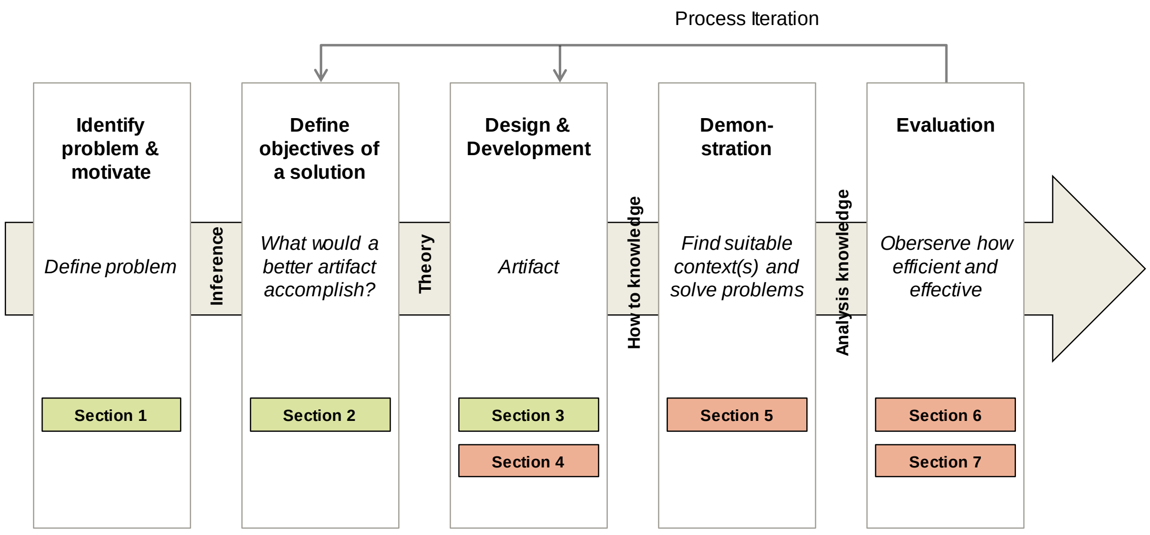


Figure 1.2.: Applied design science research methodology to solve the research questions [Pe07]

In the step *identify problem & motivate* the problem is motivated with the increasing importance of KiPs in organizations. Despite this importance software support for KiPs in existing information systems is far from being mastered. The scope of the thesis is narrowed down during the step *define objectives of a solution* by describing related work on support for highly flexible processes. While other approaches presented foundations on execution semantics for these process models, the problem of knowledge worker empowerment is still prevailing. The subsequent step in the research methodology is concerned with the *design & development* of the solution artifact. Due to its importance, this step is described in two sections in which design and development are described separately. The design includes lightweight structuring concepts for KiPs and social design principles for successful online communities that are incorporated in the design of generic user interface features.

During development the generic user interface features are implemented in a software solution that is based on the principles of a wiki. Although the research focus is on the collaborative structuring of KiPs through end-users, the implementation covers all phases of the lifecycle for KiPs. The resulting IS design artifact is applied in three application scenarios during the *demonstration* step. Main purpose of this step is to gather qualitative feedback about missing functionalities in the artifact. The feedback triggers process iterations that lead to incremental improvements of the designed artifact. The final step of the research methodology performs an *evaluation* with 145 students using the artifact to support a KiP. The main hypothesis of this thesis is evaluated through questionnaires with the participating students.

The design & development step results in the main IS design artifact of this thesis with a software solution for the collaborative structuring of KiPs through end-users. In the following we will refer to this IS design artifact with *Darwin*. To understand, execute and evaluate our research questions in a scientific way, we adhere to the seven guidelines for IS design science proposed by Hevner et al. [He04]. According to Hevner et al. [He04], this process adheres to the principle that knowledge and understanding of a design problem and its solutions are acquired in the building and practical usage of an artifact. In general, the design-oriented research approach distinguishes between the following four artifact types:

Artifact type	**Description**
Constructs	Provide a common language to describe problems and solutions that are related.
Models	Describe important aspects of the reality and enable abstraction.
Methods	Define processes and best practices that guide the solution of a problem.
Instantiations	Demonstrate that the constructs, models and methods can be implemented in a working system.

Table 1.1.: Distinct type of artifacts resulting from a design-oriented research approach

The results and findings of this thesis can be categorized based on the artifact types presented in Table 1.1. Regarding constructs our artifact provides a sound definition of the applied terminology. This terminology is reflected in the generic orthogonal structuring concepts that are used to define KiPs, e.g., data and process elements. We developed a model that describes the dependencies between the structuring concepts as well as their execution semantics. This model is designed to be domain independent and applicable in a wide range of scenarios. Our method describes the lifecycle that is required to collaboratively structure KiPs. Finally, the implemented software solution and its usage in three different use cases is an instantiation of the constructs, models and methods. We adhere to the guidelines for design science research that are presented in [He04], as described in the following.

Problem relevance Knowledge work is becoming increasingly important in organizations and software support for processes in knowledge work is still far from being mastered. In an initial step we analyze the significance of the problem and studied existing literature. The results of this initial study were presented together with a high-level solution architecture at the doctoral symposium at the RCIS conference in 2013 [Ha13a]. At a later stage our literature study on research challenges confirmed this problem relevance in the scientific community [HPM14]. Almost half of all papers on the emerging adaptive case management paradigm that aims at supporting KiPs were published in 2012 or later, which indicates the increasing interest by researchers in this area. In May 2014 the OMG released CMMN as a standard graphical modeling notation for processes in knowledge work [Ob14]. Researchers, tool vendors and organizations will increasingly adopt this standard in the future. Due to these developments as well as the increasing importance of knowledge work for organizations, that have to sustain their competitive advantage against competitors, the problem relevance is hereby approved.

Design as an artifact The constructs, models and methods developed in this thesis are implemented in a viable software solution that allows the collaborative structuring of KiPs. KiPs can also be instantiated, executed and adapted in the software solution for a wide range of application scenarios. The introduced structuring concepts for KiPs are also applicable in other collaboration platforms. Thereby, this thesis can be considered as general means for the collaborative structuring of KiPs in software solutions.

Research rigor Starting point for the research in this thesis is an extensive literature survey [HPM14] as well as use cases with research and industry partners in different domains, e.g., innovation management, requirements engineering, enterprise architecture management. Based on these findings we use metamodeling techniques, apply software engineering patterns and conduct iterative evaluation cycles to preserve rigor.

Design as a search process The main artifact, as well as its sub artifacts developed in this thesis, have gone through four iterative evaluation cycles. After conducting three use cases in an early stage of the development in the first iteration, the system is used productively for a software development project with three students during one semester in the second iteration. In a third iteration the system is used to guide four students with their master's thesis. In the final iteration the software solution is applied in a controlled experiment with 145 students for the final evaluation.

Communication of research Main findings of this research are published at international conferences and workshops after passing (double) blinded peer-review processes. Results are also presented at leading scientific communities on business process management (BPM), intelligent user interfaces (IUI) and information systems (IS). The resulting conceptual framework to support KiPs and the software solution with the required metamodel are described in detail in this thesis.

Design evaluation The software solution for the collaborative structuring of KiPs is evaluated with a controlled software experiment [Wo12]. The experiment is conducted within a lecture on web application development at the Technical University Munich with 145 participating students. During this lecture students work in teams of four students to design and implement a web application project. This project includes the development of a business model, the user interface design of the web site and the implementation of the application. Within four exercises the students have to submit deliverables and prepare presentations that will be assessed by the advisors for the final grade of the lecture. The participating 44 teams are divided into to two groups whereas of the one group serves as control group. The first group uses the software solution developed in this thesis to structure the project development process, while the second group submits the deliverables for the exercises via e-mail. Both groups are interviewed with semi-structured questionnaires after the project to assess the performance of our solution. The comparison of both groups allows the approval of our main research hypothesis that end-users without previous experience in process modeling are able to structure KiPs. The design of the mobile interface for the software solution is evaluated with a usability test within this experiment.

Research contribution Table 1.2 that is described in the following section summarizes the contributions of this thesis.

1.3. Contributions

The main contribution of this thesis is *a framework and software implementation thereof for the collaborative structuring of processes in knowledge work through end-users.* These processes are referred to as work plans that can be executed and adapted by knowledge workers while they are being executed. Work plans are instantiations of work templates that capture reusable knowledge and best practices in an organization. Adaptations made by knowledge workers in the work plans can be incorporated back in the template to enable an emergent evolution of KiPs. During the research process seven contributions were made that are necessary to fulfill this overall research goal. These contributions as well as researchers providing valuable input and evaluation partners are presented in Table 1.2.

ID	**Name**	**Brief description**
C1	Emergent structuring of KiPs	Current literature describes the need for an emergent structuring of KiPs as one of the fundamental research challenges. This is necessary due to the unpredictable characteristic of knowledge work that requires adaptations at runtime by end-users. This thesis provides a solution for this issue that empowers end-users to collaboratively structure KiPs (cf. Section 3.1). In the course of this research, we extend the Hybrid Wiki approach developed by Christian Neubert in his PhD thesis that already allows the emergent structuring of information [Ne12].
C2	Lightweight structuring concepts for KiPs that are easier to understand for end-users	End-users with limited computer science background might not be able to understand complex process modeling notations like CMMN. This thesis provides lightweight structuring concepts for KiPs as metaphors that are easy to understand for end-users and thereby allow the structuring of KiPs (cf. Section 3.2). These structuring concepts extend the previously presented concepts by Christian Neubert in the Hybrid Wiki [Ne12]. The structuring concepts are orthogonal and can be incorporated in arbitrary information systems that support knowledge-intensive tasks.
C3	Social design principles for successful online communities	Similar to the encyclopedia Wikipedia that gathers its content through crowdsourcing, processes in knowledge work also rely on contributions of individual experts. We retrieved social design principles for successful online communities and investigated existing communities for best practices (cf. Section 3.3). In the course of this research, we collaborated with Felix Michel and Yolanda Gil from the Information Sciences Institute of the University of Southern California to retrieve these principles. In the course of this collaboration the Organic Data Science Wiki is developed to support task-centered online collaborations in science [Mi15a]. In Section 2.3.8, differences between the Organic Data Science Wiki and the solution presented in this thesis are described.

ID	Name	Brief description
C4	User interface design based on social design principles	The social design principles for successful online communities are used as theoretical foundation for the user interface design of the software solution. We provide a set of user interface features that improve the engagement of knowledge workers in the processes (cf. Section 3.4). In the course of this research, we collaborated with Felix Michel and Yolanda Gil from the Information Sciences Institute of the University of Southern California to develop features for the social design principles [Mi15a].
C5	Implementation for software support of KiPs	We present the required implementation that leverages the structuring concepts for KiPs as well as the lifecycle to manage the evolution of work templates. This model provides sufficient expressiveness to model hierarchical information and process structures for a wide range of application scenarios with powerful access rights concepts (cf. Section 4.1). The base for this model is the Hybrid Wiki, which was developed in the PhD thesis of Christian Neubert. In this thesis we extend Hybrid Wikis with access rights, hierarchical information structures and a declarative process model.
C6	Interactive workbench for a subset of CMMN	Modeling experts can use the collaboratively structured KiP elements in an interactive process workbench to define declarative constraints. Dependencies between tasks and stages of a wiki page can be defined to restrict possible execution paths. The interactive workbench is based on a subset of CMMN that was recently published by the OMG as a standard for case management. This thesis provides the first implementation of this new case management standard which is based on a subset of the specification. Furthermore, our approach contributes to the existing knowledge base through its integration of end-users in the modeling of cases (cf. Section 4.2).
C7	Software support of KiPs for mobile devices	Mobile devices are becoming increasingly important in organizations. This thesis also provides a solution that is tailored to the specific needs of a mobile device (cf. Section 3.4.3). The design of the solution for the mobile device differs from the desktop version since the possible set of interactions is much different and the use cases have other goals, i.e., certain documents that require special purpose software are not changed. The solution implemented in this thesis is designed to support multiple different devices with the same server implementation. Important parts of the design and implementation of the user interface for mobile devices are developed within the master's thesis of Natascha Abrek [Ab15].

Table 1.2.: Overview of the main contributions of this thesis

With respect to existing literature, this thesis also describes four sub-artifacts that are considered as innovative contributions. In the following, these innovative contributions are explained:

1. Due to the low maturity of this research area heterogeneous terms are used by different author groups, e.g., adaptive case management, production case management, dynamic case management and case handling. A solid understanding of the core terms as well as their differences and commonalities is important for a mature discipline. Based on an analysis of existing definitions on case management this thesis presents a complete terminology to describe the field of software support for KiPs. Furthermore, characteristics of KiPs are definite against traditional highly structured workflow management solutions.

2. This thesis provides a list of research challenges that are missing in literature. Based on a structured literature review on book chapters and conference proceedings, we identified 13 challenges that are addressed in current publications on this topic. These challenges are grouped into five high-level research categories for data integration, knowledge worker empowerment, authorization and role management, theoretical foundation, as well as knowledge storage and extraction. Main focus of this thesis are research challenges related to the category knowledge worker empowerment. This category has the largest number of research challenges identified in literature and is crucial for the success of this discipline.

3. In the demonstration phase of the applied research process of this thesis, we conducted three case studies to demonstrate the usage of the developed artifact under realistic conditions. These case studies are concerned with enterprise architecture management, innovation management and requirements engineering. While the first two case studies are based on KiPs in real world enterprises, the latter case study is based on the artifact-oriented requirements engineering approach. Every case study consists of an thorough analysis of requirements for software support as well as an evaluation on the applicability of the proposed software solution. Although the primary concern of the case studies is to perform an initial evaluation of the software solution, the application is an innovative contribution since software support of processes is missing within these three domains.

4. The framework presented in this thesis is implemented in a productive web based software solution that builds on the basic concepts of the Hybrid Wiki. The software solution allows the collaborative structuring and execution of KiPs in a wiki. In addition, the software solution provides collaboration features that are valuable for the support of KiPs, e.g., a social feed showing activities in the system and personalized profile pages with the skills of the users. This sub-artifact of the thesis is considered as an innovative contribution since there are no implementations available which solve the problem of collaborative structuring of KiPs that also consider the emerging CMMN standard for the definition of processes.

1.4. Structure of the Thesis

The structure of the thesis is organized apposite to the applied design science process (cf. Figure 1.3). Every step in the design science process is described in at least one chapter. In addition, the work is organized into a framework design part and an implementation part.

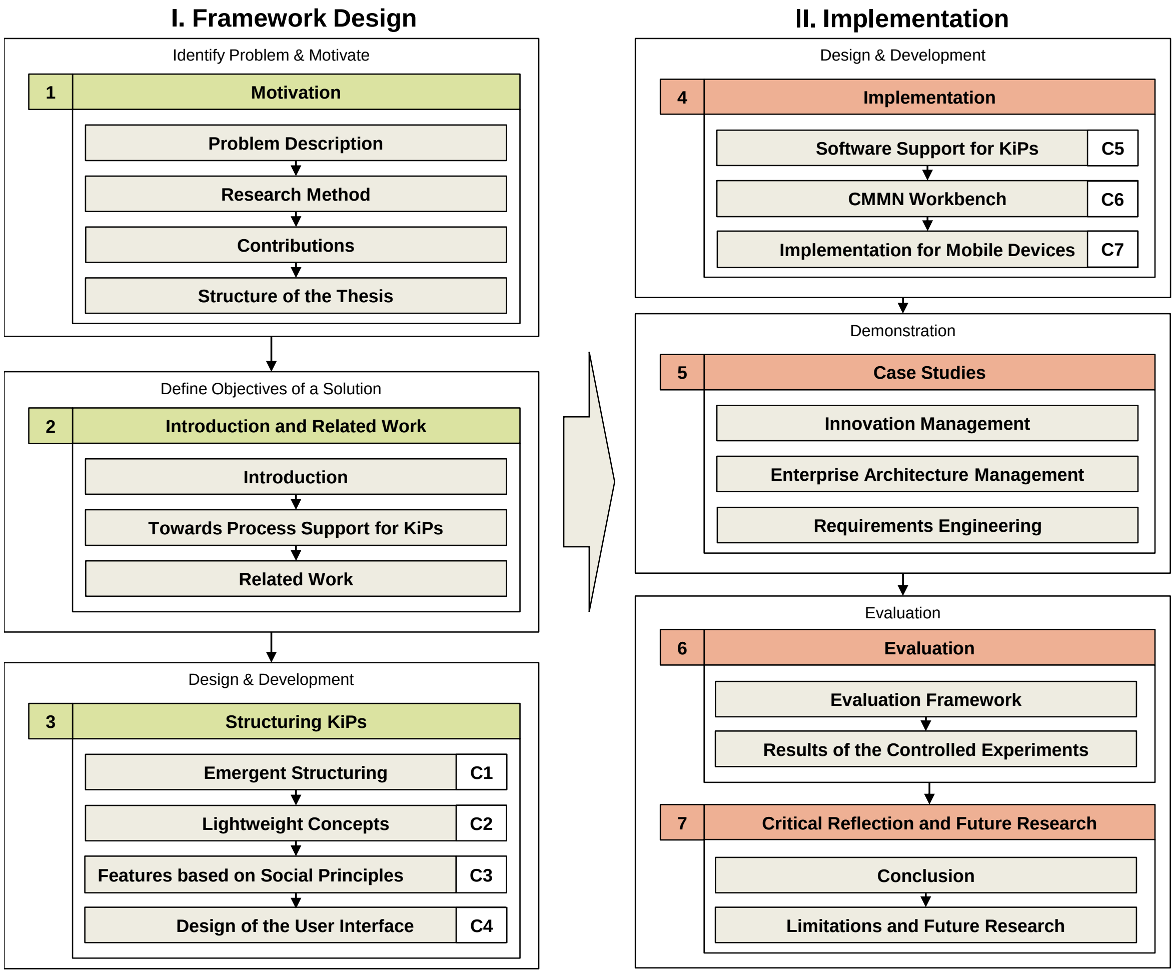


Figure 1.3.: Outline of the thesis linked to the contributions, cf. Table 1.2 and Figure 1.2

The framework part continues with an introduction and related work, which also contains a summary on requirements for software support of KiPs. Based on these requirements the conceptual solution for the structuring of KiPs through end-users is described. The design of the framework consists of the emergent structuring, lightweight structuring concepts, features based on social principles and the user interface design. In the second part the implementation of the framework is presented with the detailed description of the software solution. This implementation is demonstrated in three use cases for innovation management, enterprise architecture management and requirements engineering. Within the implementation a CMMN workbench and an implementation for mobile devices is presented. Finally, the evaluation is presented based on a controlled software experiment with 145 participants before the thesis concludes with a critical reflection and future research outlook.

The most valuable assets of a 20th-century company were its production equipment. The most valuable asset of a 21st-century institution, whether business or non-business, will be its knowledge workers and their productivity.

Peter Drucker

CHAPTER 2

Introduction and Related Work

This chapter provides theoretical foundations on three disciplines and related literature for this thesis. These three disciplines are integrated in the framework that is developed for the software support of KiPs. The first discipline is process support in knowledge work that is introduced in Section 2.1.1 with some fundamental definitions on knowledge, knowledge work and knowledge-intensive processes. In the second discipline the integration of data and processes, which is an important aspect of KiPs, is presented in Section 2.1.2. In Section 2.1.3, the third discipline on successful online communities that provides valuable concepts that are integrated in this research is presented. Based on these three disciplines generic requirements for software support of KiPs are derived in Section 2.2.1. These requirements are the basis for the conceptual framework and the software solution developed in this thesis. In Section 2.3, the literature analysis approach and the identified results related to the solution described in this thesis are presented.

2.1. Introduction

Before KiPs are defined some fundamental definitions on the terms knowledge and knowledge work are described briefly based on the literature study conducted in [MKR13]. Building on a common understanding of the term knowledge, we introduce the dynamic theory of organizational knowledge creation. This theory is important for this thesis since it describes how organizations can articulate and amplify knowledge created by individuals in an organization. The definition of KiPs is presented along with the four key characteristics of these processes. Many approaches for highly structured processes have been presented in current literature, whereas these solutions are not suitable for the characteristics of KiPs. KiPs are compared against other more structured processes based on the process management spectrum.

2.1.1. Knowledge-Intensive Processes

At the beginning, the notion of knowledge needs to be delineated from the terms data and information, which are often used synonymously in the computer science discipline. Data represent a set of symbols on a syntactic level without meaning. Information extends data with interpretation and therefore describes data with meaning and semantics [AN95]. Based on these definitions, knowledge can be described as information in the context of an agent's reasoning resources. For this thesis it is particularly important to notice that knowledge can also be concerned with processes and routines describing what has to be done to achieve some organizational goals. Davenport et al. [DP98] define knowledge work as follows:

Definition: Knowledge
Knowledge is a fluid mix of framed experiences, values, contextual information and expert insights that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organizations, it often becomes embedded, not only in documents or repositories, but also in organizational routines, processes, practices and norms.

Embodied in this definition of knowledge work is the distinction between tacit and implicit knowledge made by epistemological scientist Polanyi [Po66]. Tacit knowledge can be hold by humans although they are not able to explicitly communicate this knowledge. Explicit knowledge can be communicated through a formal language or information with respect to the previous explanation of information. Based on this distinction of explicit and implicit knowledge Nonaka presented the theory of organizational knowledge creation [No94]. Understanding how organizations create knowledge is crucially important in the new economy of the 21st century. In this economy knowledge is more than just another resource alongside the traditional production factors labor, capital and land. Knowledge is becoming the only considerable resource that determines how successful organizations are in our society [Dr93]. Main reason for this is that knowledge creation is the foundation for continuous innovation in organizations, which in turn leads competitive advantage against competitors [NT95]. Figure 2.1 illustrates how knowledge is created in an organization according to Nonaka's theory.

Organizational knowledge creation can be described with the continuous transformation between tacit and explicit knowledge. This conversion takes place in four subsequent sequences that are passed through in the knowledge creation spiral. During the *externalization* implicit knowledge that is available in the organization is gathered and documented. In this thesis we are interested in knowledge that is concerned with processes and routines in an organization. In the subsequent *combination* sequence this external knowledge is combined with already existing explicit knowledge. New explicit knowledge is made implicit in an organization during the *internalization* sequence. This is usually done by providing documentations or training employees in an organization. In our context internalization provides employees with new processes or routines. Finally, the internal knowledge is applied in the organization during the *socialization*. During this socialization the knowledge is spread in the organization and leads to new knowledge gains than can be externalized. With every cycle organizations create new explicit and implicit knowledge, which is the foundation for innovation and competitive advantage. This new knowledge originates from individuals and is transformed during the knowledge creation process to groups and the entire organization.

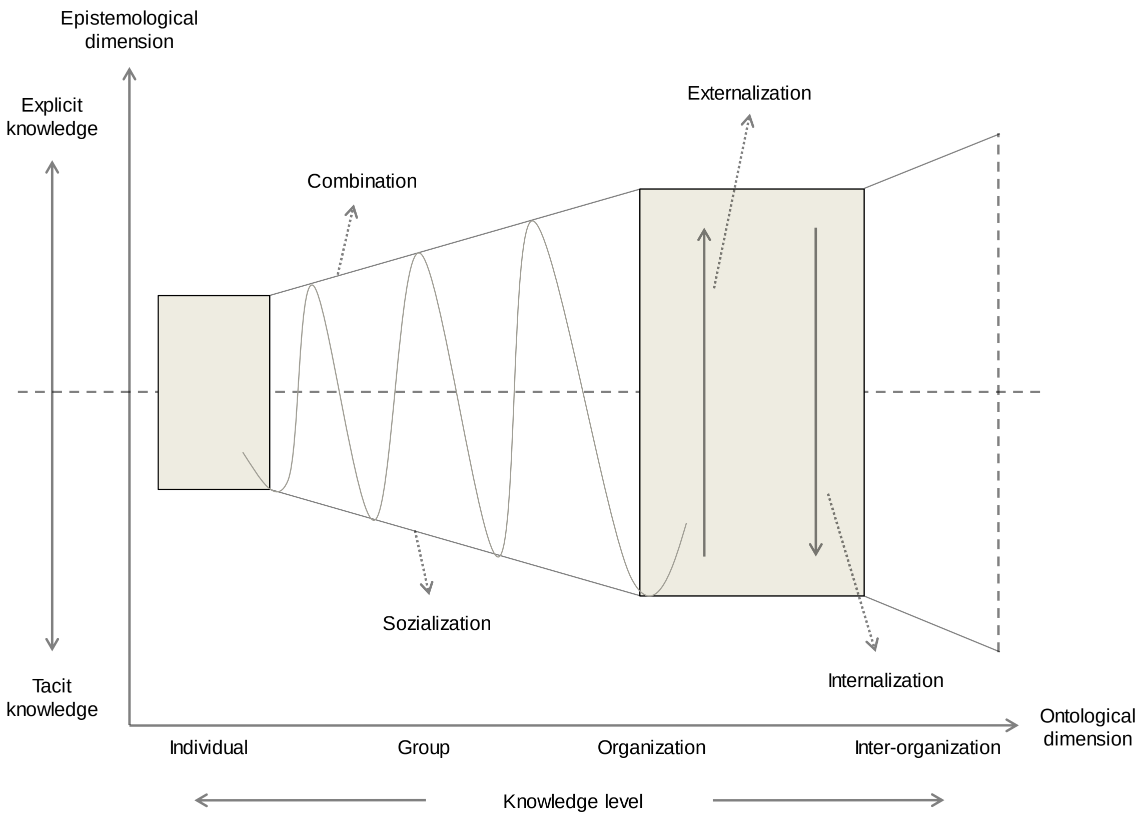


Figure 2.1.: Knowledge creation spiral for organizations introduced by Nonaka [No94]

Initial studies that are related with knowledge work studied changing working environments that have a much higher degree of mental activities. Among these studies are publications concerned with the transformation of the industrial society described by Bell [Be73] and Machlup [Ma62]. In his PhD thesis Hube analyzed existing fundamental definitions on knowledge work that are briefly summarized in this thesis [Hu05]. Drucker defined a knowledge worker as *"An employee whose major contribution depends on his employing his knowledge rather than his muscle power and coordination, frequently contrasted with production workers who employ muscle power and coordination to operate machines"* [Dr77]. In subsequent publications Drucker emphasized the importance of innovation for successful knowledge work. Although his work provided important foundations for the definition of knowledge work, the descriptions remain on a rather abstract level without detailing differences of work types. Beruvides et al. differentiate work into the three types blue collar work, white collar work and knowledge work [BS87]. The blue collar worker can be described as traditional factory workers, while the white collar workers are working in the service industry. The third type knowledge work is characterized by four criterions. First, the input of knowledge work is difficult to determine and it often has no directly measurable impact on the output. Second, the work can be conducted mentally or manually. Third, the output of knowledge work is mainly intangible. Fourth, in contrast to blue collar workers that have no authority to dispose, knowledge workers are able to make own decisions.

While early definitions of knowledge work are closely defined with professional groups, subsequent publications tried put a stronger emphasize on the work character. Scarbrough describes knowledge work as comparatively unstructured and organizationally induced [Sc99]. In contrast to previous publications that describe knowledge work based on defined methods for jobs, these characteristics capture the changing demand of companies. He identified several developments alongside this increasing demand for knowledge workers that include the collapse of traditional working model. The importance of knowledge work is increasing in many professional groups and the establishment of new sectors in the economy that create knowledge. Furthermore, information technology is an important factor for the codification and transportation of knowledge. Davenport distinguishes between several types of knowledge work that can be classified along the two dimensions complexity of work and level of interdependence [Da05]. Knowledge work having low level complexity that is conducted by individuals is called transaction in this classification. Individuals that perform complex work based on judgment conduct expert knowledge work. Knowledge work with a high level of interdependence can be either integration or collaboration depending on the complexity of work. This classification is rather broad and generic making it difficult to retrieve an exact definition for knowledge work.

An assessment of existing definitions for knowledge is described by Hube in his PhD thesis [Hu05]. The assessment is performed with four criteria that are based on the applicability for labor science, description of the knowledge work process, individuality and selectivity against routine work. All of the definitions found in literature lack selectivity against routine work and description of the knowledge work process. Many of the definitions are too generic and the missing description of the knowledge work process makes them not suitable for the scope of this thesis. Hube introduces a more specific definition for knowledge work that incorporates these criteria. Throughout this thesis, we will rely on this definition of knowledge work that has been translated to English in [MKR13]:

Definition: Knowledge Work
Knowledge work is comprised of objectifying intellectual activities, addressing novel and complex processes and (work) results, which require external means of control and a dual field of action.

In the first part of the definition knowledge work is delimited against intellectual work. Knowledge work excludes routine mental work and only addresses novel and complex processes as well as work results. In the second part of the definition knowledge work processes are divided into two fields of action, i.e., the referential field of action and the actual field of action [Re88]. During the referential field of action activities of knowledge workers are mainly concerned with theoretical considerations. In this field of action different approaches can be evaluated and planned without causing any direct impact on the problem to solve. The management of complex processes takes places during the actual field of action with the required resources and activities that are necessary to produce the desired outcome. This common understanding of the term knowledge work is the foundation for the definition of KiPs that is presented in the following. From a theoretical perspective KiPs are within the intersection between knowledge management (KM) and BPM. While current BPM approaches manage processes and process-related knowledge separately, KiPs requires a tighter integration between both elements to facilitate organizational knowledge creation.

Before the definition of KiPs is presented one needs to detail intellectual activities that take place in knowledge work. Intellectual activities are performed by humans, who create, share, transfer and apply knowledge in the context of their processes. Purpose of these processes is the achievement of organizational goals and the creation of value [MF11a]. This means that intellectual activities are concerned with the creation of completely new knowledge and the application of already existing knowledge. Taking these considerations into account, we rely on the following definition for KiPs provided by Vaculin et al. in [Va11] throughout the thesis:

Definition: Knowledge-intensive Processes
Knowledge-intensive processes (KiPs) are processes whose conduct and execution are heavily dependent on knowledge workers performing various interconnected knowledge-intensive decision making tasks. KiPs are genuinely knowledge-, information- and data-centric and require substantial flexibility at design- and runtime.

In [MKR13], four key characteristics for collaborative knowledge work are described with (i) uncertainty, (ii) goal orientation, (iii) emergence and (iv) growing knowledge base. Knowledge workers are making autonomous decisions and collaborate on tasks depending on knowledge artifacts. These tasks are hard to predict and it is not possible to predefine them into a classical control-flow because of their uncertainty. Goals are used by knowledge workers to align their activities and resources. During the course of actions processes gradually emerge through alternating planing and operative working phases. In the operating phases knowledge artifacts are created with intermediate results that contribute to a growing knowledge base.

Processes can be classified along the degree of structure and predictability, which directly influence the possible level of modeling, control and automation. Figure 2.2 illustrates this process management spectrum according to Di Ciccio et al. in [DCMR13]. KiPs are mainly located at the two layers at the bottom with unstructured and loosely structured processes. KiPs require a high degree of flexibility and are unpredictable by nature. At the top of the spectrum shown in Figure 2.2 *structured processes* describe routine work that is highly predictable in advance. The required flexibility is rather low and logic in the processes can be predefined before instantiation through dedicated process designers. End-users are usually not able to influence the course of action through their decisions at runtime. Process logic contains activities, their dependencies and resources that perform activities in the processes. As a result all options and decisions can be captured in a process model a priori. The process model can be repeatedly instantiated with little variations during the execution of processes. Examples for structured processes are production and administrative processes.

Similarly, *structured processes with ad-hoc exceptions* have related characteristics with activities that follow a predefined plan. Main difference to structured processes is that external events and exceptions can change the structure of the process at runtime. Process adaptation strategies might be required to change the predefined work practice. Anticipated adaptations might be predictable and defined in advance in the process model. In *unstructured processes with pre-defined segments* the overall process logic is not explicitly defined and work practices are mainly ad-hoc. Nevertheless, some fragments of the process can be predefined and structured based on regulations. The selection and composition of process fragments usually depends on the specific case and might change with every instantiation.

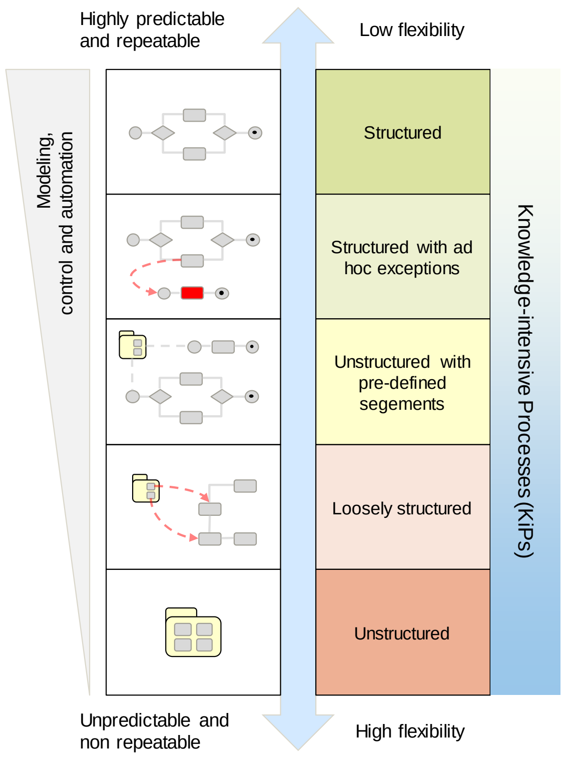

Highly predictable and repeatable
Low flexibility
Modeling, control and automation
Structured
Structured with ad hoc exceptions
Unstructured with pre-defined segements
Loosely structured
Unstructured
Knowledge-intensive Processes (KiPs)
Unpredictable and non repeatable
High flexibility

Figure 2.2.: Process management spectrum [DCMR13]

Loosely structured processes cover a wide range of work practices in which process logic is not defined prescriptively through procedures. Possible execution paths are narrowed down through business rules in a declarative manner. In these loosely structured processes the set of required actions might be foreseeable, but the exact execution order cannot be determined due to many possible alternatives. Business rules are specified with constraints that implicitly describe these alternatives by prohibiting undesired execution behavior. Recently, many approaches for these declarative process models have been proposed to enable this flexibility [Va11]. Decisions about which actions to take next are made autonomously by experts participating in these processes.

At the bottom of the process spectrum *unstructured processes* are unpredictable and require the highest degree of flexibility. In contrast to loosely structured processes, the set of actions and their execution order cannot be determined in advance. Process participants make all decisions autonomously and the structure of the process thus typically evolves dynamically during enactment. As a result only very little automation can be provided for these processes that directly represent knowledge work. The structure of the processes also varies for every case and process participants have to handle unexpected exceptions. Only the goal of the unstructured process is usually known in advance. Due to this uncertainty, many decisions have to be made by knowledge workers during the process execution. This requires simple structuring concepts that do not overwhelm end-users without previous knowledge in process modeling.

2.1.2. Data and Processes

Process support for knowledge work requires means to deal with unpredictable situations at runtime and an integration of data in the processes. Traditional process modeling is divided into a data and behavior modeling part to reduce complexity. These *activity-centric approaches* are focused on highly structured and predictable processes. In activity-centric approaches atomic activities are explicitly defined with their relationships in an imperative and procedural way. As a result activity-centric processes always follow a pre-specified order of their tasks allowing only little deviations at runtime. Another limitation of activity-centric approaches regarding their applicability for knowledge work, is that data on business objects and their attributes is mostly unknown to the process engine. This data is typically managed separately to the process engine by database applications and enterprise systems.

KiPs cannot be represented with a set of activities with predefined precedence relations. In KiPs the availability and status of data objects are the main driver for the execution of the process. Various *data-centric approaches* have been proposed to model processes with a tight integration of processes and data [KR12b, KWR11]. Figure 2.3 illustrates a comparison between activity- and data-centric approaches. In an activity-centric approach users only have a view on their worklist that contains tasks for a given point in time, while the link to data being manipulated in the processes is missing. Services that are used in the processes are typically hard-coded in activity-centric approaches. Users in data-centric approaches require an integrated view on data, services and processes with services that are generic. Users should be able to switch between these different views to complete knowledge-intensive tasks in a natural way. Main advantage of this approach is that a flexible execution of unstructured and knowledge-intensive processes becomes feasible.

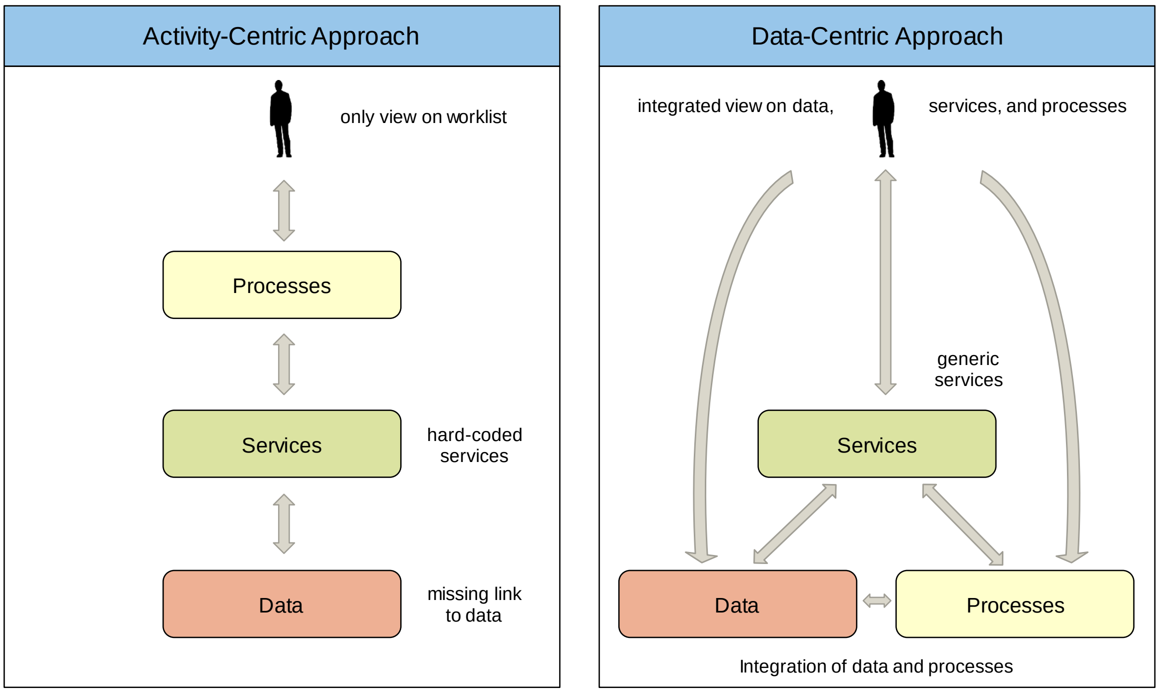


Figure 2.3.: Comparison between activity- and data-centric approaches [KR11]

The simplest solution for the specification data-centric processes are finite state machines [Ku03, Re09, KWR11]. In this approach the behavior of information objects is defined with a state machine and transitions between these states specify possible actions. According to Marin et al. [MHV13], an important ingredient in enabling flexibility in data-centric approaches is the shift from procedural to declarative lifecycle models. In the late 1990s the data-centric process framework Vortex has been introduced [Hu99]. It uses condition-based guards to control when certain modules are launched. Due to limitations in the core concepts the use for general-purpose BPM is limited. The artifact-centric approach based on the declarative Guard-Stage-Milestone (GSM) model originates from that work to support more flexible data-centric processes [Hu11a, Hu11c, DHV13]. In this approach the process logic and the data layer are conceptually integrated in one concept that is called artifact. In the artifact-centric approach processes are represented with collections of artifacts that encapsulate data and a lifecycle. Although the artifact-centric approach was developed for traditional BPM domains, it has emerged as suitable for case management [MHV13]. According to Hull et al. [Hu11a] artifacts are defined as follows:

Definition: Artifacts
Key conceptual entities that are central to the operation of a business and that change as they move through the business's operations. An artifact includes both an information model to capture all business relevant data and a lifecycle model that specifies the possible ways it might progress through the business.

Among the key conceptual entities are *stages* that are used to cluster tasks that are performed within an artifact. In contrast to approaches based on state-machines stages can run parallel. *Milestones* are used to represent intermediate business goals that an artifact may fulfill. These milestones are tightly linked to stages in the GSM model. *Guards* describe when a stage is opened and can be entered. Once a stage is opened and all associated milestones are true, the stage automatically closes since the purpose of the stage is achieved. Decisions whether guards and milestones are activated are described by *sentries*. Sentries can be specified based on the event condition action (ECA) structure. It basically states that actions are triggered after an event occurs and the condition is fulfilled. Either the event or condition part might also be empty and the action can still be triggered. The action in this context is the activation of the attached guard or milestone. Once an incoming event is registered in the system, all ECA rules that are applicable to this event are fired.

This GSM model and previous works related to data-centric business processes provide the foundation for CMMN that is introduced in Section 2.3.7. The notation consists of tasks, stages, milestones and events similar to the GSM model. Main purpose of CMMN is to provide a modeling notation standard for case management. It avoids describing an implementation but provides a notation, metamodel, interoperability between tools and minimum execution semantics. CMMN and the GSM model only differ in two major design decisions. First, milestones are not tightly coupled to stages in CMMN since they can be defined separately from stages. Second, CMMN allows to make adaptations on the runtime model which is not considered in publications related to the GSM model. The notation might be useful for requirements of KiPs that are related to modeling. Aspects concerned with the user interface, facilities for end-users, collaboration and emergence have to be provided by an appropriate execution environment.

2.1.3. Successful Online Communities

Processes can be divided based on the process management spectrum that is shown in Figure 2.2. Processes at the top of the process management spectrum are highly predictable and repeatable. They include processes that are structured and structured with ad-hoc exceptions. Due to their high degree of structure these processes can be defined in a top-down manner by few modeling experts that are familiar with the process or collaborate closely with domain experts. KiPs cannot be completely predefined in a top-down manner by modeling experts due to their unpredictable characteristic. Instead, these processes emerge bottom-up through contributions of individual knowledge workers. Every participating knowledge worker needs to bring in his expertise and experience to accomplish the overall goal of the KiP. Many knowledge-intensive problems are so complex and elaborate that they require a large amount of highly-specialized participating experts that contribute with their knowledge. For all those reasons it is critically important that knowledge workers are not only empowered to participate in the structuring of their processes, but it is equally important that they are also willing to do so. Figure 2.4 illustrates the evolution of scientific work over the last hundred years. Although this example is based on scientific work, similar developments can be observed for many other application domains, e.g., open source software development.

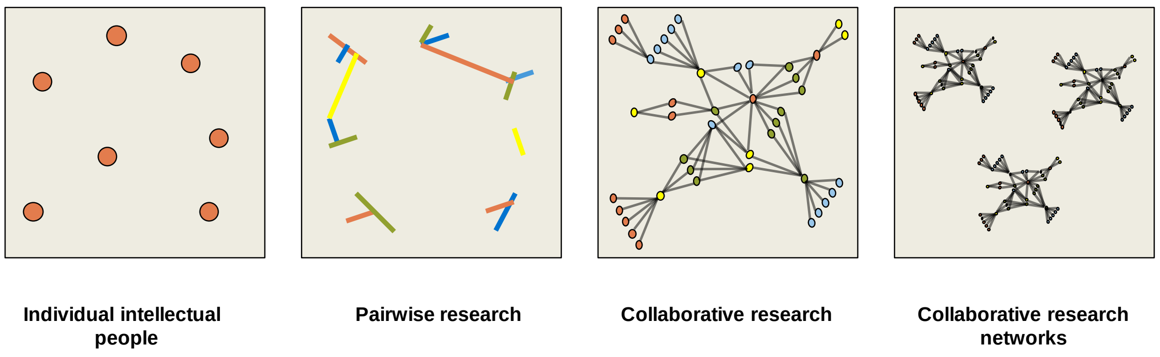

Individual intellectual people
Pairwise research
Collaborative research
Collaborative research networks

Figure 2.4.: Evolution of scientific work over hundred years according to Michel [Mi15a]

In the beginning individual intellectual people published pioneering publications, e.g., Albert Einstein, Isaac Newton and Charles Darwin. In the second phase scientific work has been conducted through pairwise research, e.g., James D. Watson and Francis Crick unscrambling the DNA's structure. Collaborative research emerged in the subsequent stage with many researchers working together, e.g., the human genome sequencing consortium. Today large scientific projects are networks of several collaborative research groups, e.g., the ATLAS project at the large hadron collider in CERN or the ENCODE project. Over the past hundreds years science has become much more collaborative and the next step of this development is the establishment of online communities. With these communities external workers that are not part of the participating research groups or organizations are engaged. These external workers can bring in valuable knowledge or resources related to the research goal. In many domains numerous examples of online communities have proven how successful virtual teams can collaborate through technology platforms, e.g., forums, blogs, wikis and networking sites. Similar to offline communities, their digital counterparts support their members with information sharing, social support and companionship.

Although several examples demonstrated the potential benefits of online communities, many approaches struggle to become successful because they are not able to attract enough active members. Only after a critical mass of members that contribute content is reached, sufficient content is produced that attracts new members to the community. Figure 2.5 illustrates critical design challenges that have to be solved for online communities to become successful. Based on the findings of Kraut and Resnick that are presented in [Kr12a], five critical design challenges for successful online communities are presented in the following.

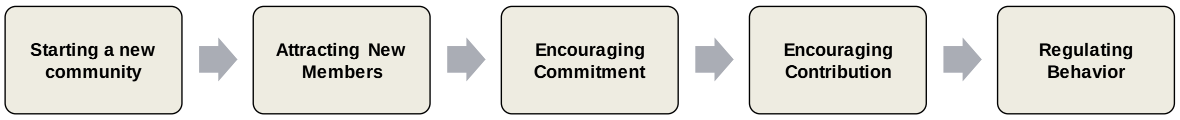


Figure 2.5.: Design challenges for successful communities according to Kraut et al. [Kr12a]

Communities become successful because they have a large amount of content that is valuable for other people, e.g., videos, software source code, articles, or questions and answers. *Starting a new community* is the first critical challenge since this content is not available from the beginning and needs to be created first. Main reason for this challenge is the critical mass problem, which means that the community doesn't have enough content to attract new members and therefore insufficient new content is created. Once the community has been established, it is important to *attract and socialize new members* for the community to grow or to replace members who leave. Many communities need to identify and encourage members that have certain skills as well as motivation to contribute. New members of a community have to be socialized and need to learn appropriate behavior in the community.

Encouraging commitment among members of a community improves their willingness to stay in the community and contribute to it. People in offline and online communities that are more committed to an organization are more satisfied, productive and less likely to leave [MZ90]. Especially online communities have to deal with the challenge of commitment since changing to another community is much easier than in the offline world. In the offline world changing or leaving a community is often related with loss of salary, job status, or geographic proximity. The existence of a community depends on its content and therefore members of a community have to *encouraged to contribute* with their knowledge. Types of contributions could be code in software projects, videos, tutorial, or solutions to problems. Communities in which members contribute with their knowledge are much more likely to become successful.

Once a community is established and has a large number of members, it is important to avoid arising conflicts or provocation from other members that might have conflicting interests. These situations require mechanisms to *regulate behavior* of members in a community. Many of these challenges only arise in online communities and not in real world organizations. Main reasons for inappropriate behavior in online communities are the anonymity of members, ease of entry and exit and textual communication. The anonymity of online solutions lowers the threat of social accountability. Due to the ease of entry and exit in online communities members are less tied to the community and this limits the effect of sanctions. Finally, textual communication is prone to misinterpretation because of missing nonverbal gestures.

2.2. Towards Process Support for Knowledge Work

This section presents fundamentals towards process support for knowledge work with generic requirements and three sample application scenarios. The requirements are grouped into seven distinct categories related to (i) data, (ii) knowledge actions, (iii) rules, (iv) goals, (v) processes, (vi) knowledge workers and (vii) environment [DCMR13]. General characteristics that are common for all KiPs are illustrated with the conceptual diagram in Figure 2.6. Depending on the specific application scenario of the KiP, additional requirements might have to be addressed, e.g., interfaces to medical devices in the healthcare domain. Due to the plethora of possible application scenarios this section only covers generic requirements.

Knowledge workers are usually involved in several work plans at the same time that are used to structure their processes (v). For every work plan knowledge workers require an overview about the current progress of tasks (ii) and information elements (i). Tasks in the work plan might impose logical and temporal dependencies defined through rules (iii). Work templates describe best practices for one particular context that can be reused and adapted if necessary by the knowledge workers (vi) to accomplish certain goals (iv). Work plans need to be integrated with their environment (vii) since knowledge workers often use dedicated special purpose applications, e.g. integrated development environment or computer-aided design.

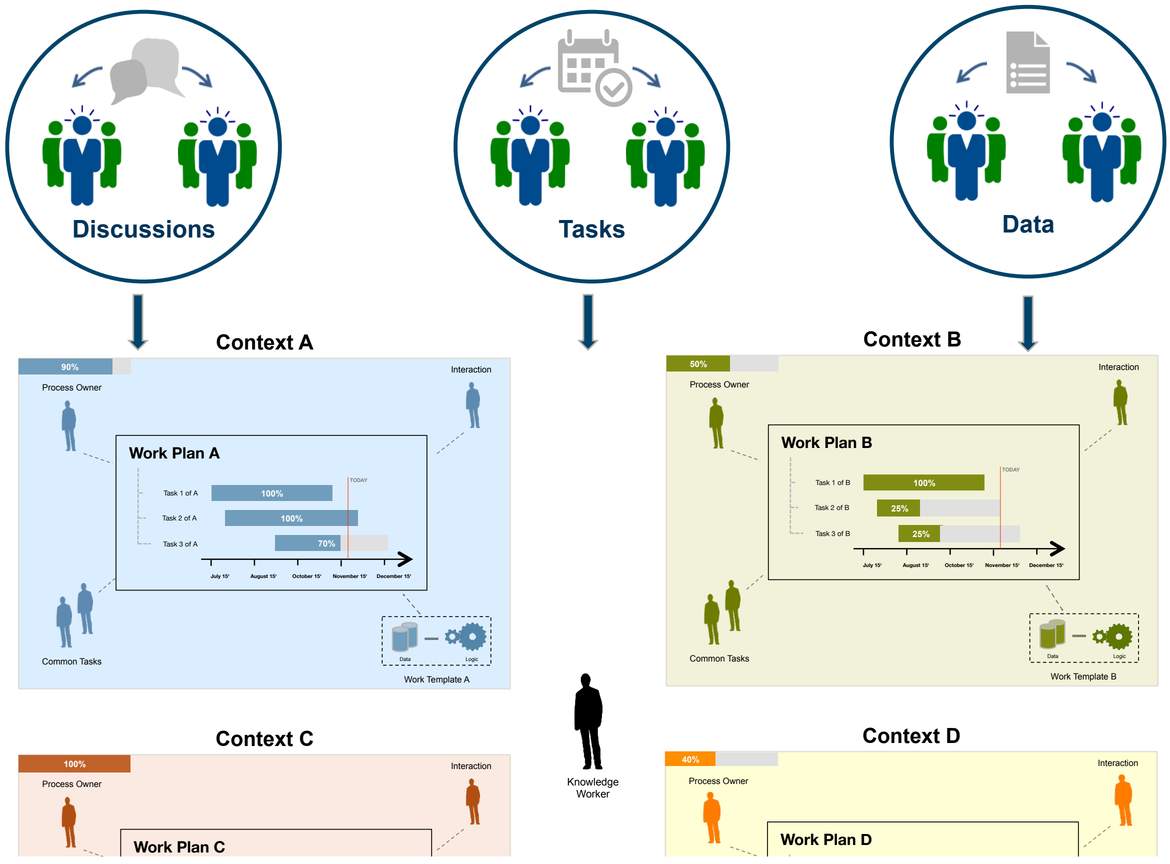


Figure 2.6.: Knowledge workers participate in KiPs in various different contexts [Mu12]

2.2.1. Generic Requirements for Knowledge-Intensive Processes

In this section, current research on requirements for KiPs is presented as foundation for the framework and software solution presented in this thesis. Requirements for KiPs found in literature are mainly gathered from real-world application scenarios from different domains, e.g., healthcare, product development, criminal investigations and court case management. At this stage generic requirements for KiPs are gathered from current literature and their relevance for the three application scenarios is investigated. In the following subsection these three application scenarios for KiPs are described to improve the understanding of the generic requirements. The requirements are presented along with the structure introduced by Di Ciccio et al. [DCMR13] in seven groups. All requirements are summarized and consolidated from recent publications by Hauder et al. [Ha14a], Mundbrod et al. [MR14] and Di Ciccio et al. [DCMR13]. We decided to select these references since they are from three distinct research groups that studied requirements for KiPs independent from each other.

In addition to these generic requirements some application scenarios might have domain specific requirements that cannot be covered in this section due to the large number of potential scenarios. Characteristics of KiPs appear in a variety of different application scenarios in organizations. Due to this wide range of KiPs the relevance of these generic requirements might be diverging from scenario to scenario. Table 2.8 illustrates the relevance of the requirements for the application scenarios investigated in this thesis. For every requirement we distinguish between three different levels for every scenario. Highly relevant requirements are explicitly mentioned for the scenario, i.e., without this requirement the scenario cannot be adequately supported. Requirements with medium relevance are not explicitly stated in the scenario, but they can be derived from actual use cases that were observed. Requirements with a basic relevance could not be explicitly or implicitly derived within the particular use case.

Integration with data is an important requirement for software support of KiPs. In activity-centric approaches the link to data is missing, i.e., data is only manipulated through forms that are available during the activities. Within KiPs knowledge workers need to be able to collaboratively access and manipulate data at any time in the processes. This is only possible with a suitable information model that structures the data appropriately. An appropriate information model combines structured and unstructured information, i.e., files, strings, integers, dates, enumerations and complex types like ideas. Due to the unpredictable characteristic of KiPs it is important that these information structures can be adapted by end-users. Table 2.1 summarizes requirements for KiPs that are related to data.

Name	Description	Source
R1 Data modeling	All data that is manipulated by the process has to be stored in an information model. This information model needs to support different levels of abstraction and arbitrary data types.	[DCMR13]
R2 Late data modeling	New knowledge that arises at runtime of the process may involve the creation and modification of data. Knowledge workers have to be able add new data to the information model as well as alter or remove the existing ones.	[DCMR13]

Name	Description	Source
R3 Access to appropriate data	All data in the information model has to be accessible to the knowledge workers at any time and not only while certain data for an action needs to be manipulated. This access to data needs to consider appropriate authorization rights.	[DCMR13]
R4 Synchronized access to shared data	Many different knowledge workers might manipulate the same tasks and data concurrently. Consistency during concurrent execution needs to be ensured.	[DCMR13]
R5 Data model comprehensibility	Every knowledge worker should be able to make adaptations on the data model. This requires a limited number of entities in the data model without loosing the right balance between expressiveness and comprehensibility.	[MR14]

Table 2.1.: Name, description and source of data requirements for software support of KiPs

In an organization work needs to be coordinated and allocated among many knowledge workers that perform knowledge actions autonomously. These knowledge actions have to be represented through data-driven tasks in a software solution. Compared to workflow management solutions that aim at automating as many steps as possible in a process, tasks in KiPs usually consist of intellectual activities that are performed manually. The structuring of KiPs allows to capture best practice knowledge about procedures and guide workers in an organization. Furthermore, additional roles compared to workflow management are required that enable more flexibility, e.g., to repeat tasks during an approval procedure or to skip tasks that are not necessary. Knowledge action requirements are described in the following table:

Name	Description	Source
R6 Represent data-driven actions	Actions in KiPs depend on the evolution of the information model to support data-driven progression of the process. Constraints defined on data in the KiPs guide the execution of knowledge actions.	[DCMR13]
R7 Late actions modeling	Due to the emergent characteristic of KiPs, actions have to be added and manipulated by knowledge workers during process enactment. This requirement can be combined with late data modeling to associate actions with required data.	[DCMR13]
R8 Assignment of knowledge action roles	In addition to execution roles that are known from workflow management and used to assign responsible persons for knowledge actions, new roles for skip, delegate and redo are necessary. These additional roles specify which knowledge workers are allowed to perform these actions.	[Ha14a]

Name	Description	Source
R9 Late assignment of knowledge action roles	The assignment of knowledge action roles might have to be adapted or new knowledge workers have to be added or removed to roles during process enactment. This requirement can be combined with late actions modeling.	[Ha14a]

Table 2.2.: Name, description and source of knowledge action requirements for KiPs

Unstructured and loosely structured processes require a declarative process model that narrows down permitted tasks during the execution of a KiP. Instead of specifying one predefined and standardized path, declarative process models only exclude certain sequences that are not allowed [Pe08]. Main difference between both paradigms is that knowledge workers have more freedom and can make own decisions on the most suitable path in the KiPs. The degree of freedom varies with the number of constraints that are added to the KiP, i.e., the flexibility is limited with an increasing number of constraints. Furthermore, making adaptations in a declarative process model can be considered as easier since new constraints can be added without an understanding of the entire network of tasks. In highly structured processes new tasks have to be integrated in an existing network, which might be very complicated and difficult to understand for end-users. For this purpose the requirements for the definition of rules and constraints in KiPs are described in Table 2.3.

Name	Description	Source
R10 Formalize rules and constraints	The execution of KiPs needs to be constrained by policies and rules, e.g., producer and consumer pattern in which tasks are enabled after certain other tasks are completed. Knowledge workers have to be able to explicitly define constraints on tasks and data.	[DCMR13]
R11 Late constraints formalization	Similar to late knowledge actions and data modeling that were described earlier, knowledge workers must be able to formalize new constraints during the process enactment.	[DCMR13]

Table 2.3.: Name, description and source of rules and constraint requirements for KiPs

According to the definition of KiPs, goal orientation is another main characteristic that delineates them from highly structured and predictable processes. Goals are typically created in an early stage of a KiP and constantly refined during process execution. In many application scenarios goals are refined through milestones and quality gates that are defined early in the process. In these situations they are used to ensure high quality and standards, e.g., in engineering processes [MR14]. In other scenarios static milestones are not used and dynamic sub-goals are created, e.g., through physicians in healthcare processes [MR14]. The fulfillment of goals depends on the completion of tasks or the status of data elements. Compared to knowledge actions that change during process enactment, goals that are located on the highest level in a KiP remain rather stable during the lifetime of the process. In many situations goals are the only input that is available in the beginning and knowledge workers fulfill them in a self-organized and autonomous working mode. An approach that aims to support KiPs needs to fulfill the following requirements related to goals:

Name	Description	Source
R12 Goals modeling	Mechanisms for representing one or more process goals defined on data are required. Concrete goals can be defined and their fulfillment may be associated to the result of data elements. Similar to knowledge actions, goals have to be organized in hierarchies and they might be decomposed during process enactment.	[DCMR13]
R13 Late goal modeling	New goals might arise as a result of decisions made by knowledge workers or due to the evolution of data elements. Knowledge workers must be able to associate new goals during process enactment or change existing goals. New goals are typically created as milestones or sub-goals depending on the concrete application scenario.	[DCMR13]

Table 2.4.: Name, description and source of goal requirements for KiPs

Data, knowledge actions, constraints and goals are fundamental building blocks that can be found across all KiPs. The combination of these building blocks in processes results in new requirements that have to be supported by an approach for KiPs. While some of these elementary building blocks might be supported by existing solutions, the following requirements result from an integration of the building blocks:

Name	Description	Source
R14 Support for different modeling styles	KiPs are a combination of different knowledge entities, e.g., data and tasks. The modeling of these entities should not be divided and therefore various modeling alternatives have to be provided.	[DCMR13]
R15 Visibility of the process knowledge	An aggregated view on data, tasks, constraints and goals for a running process needs to be provided. This is necessary so that knowledge workers can monitor the progress.	[DCMR13]
R16 Flexible process execution	New goals might arise as a result of decisions made by knowledge workers or due to the evolution of data elements. Knowledge workers must be able to associate new goals during process enactment or change existing goals.	[DCMR13]
R17 Deal with unanticipated exceptions	The unpredictable nature of KiPs requires manual or automatic procedures that deal with unanticipated exceptions. These procedures mainly depend on the specific case.	[DCMR13]

Name	Description	Source
R18 Migration of process instances	As a result of unanticipated exceptions data and tasks in KiPs often evolve over time. Running instances of KiPs have to be updated with these changes at runtime through migrations.	[DCMR13]
R19 Learning from event logs	Event logs that capture traces of previously executed processes have to be used to learn KiPs. This mechanism can be used for discovering and improving the structure of a KiP.	[DCMR13]
R20 Learning from data sources	In addition to event logs that capture traces of process executions, data sources can be used to discover and improve the structure of a KiP. These data sources might cover a broad range of data from unstructured, semi-structured, to structured texts and files.	[DCMR13]
R21 Recommendations during process enactment	During the process enactment knowledge workers have to be supported with the adaptation of work plans through recommendations. Recommendations can be related to tasks and data elements that were used in similar situations.	[MR14]

Table 2.5.: Name, description and source of process requirements for KiPs

Process support for knowledge work goes beyond simple automation of repeated activities. KiPs coordinate work among knowledge workers that have dynamically changing roles and responsibilities. Requirements that are related to knowledge workers in the processes are described in the following table:

Name	Description	Source
R22 Knowledge workers' modeling	KiPs require a resource model consisting of participants with roles and capabilities. Roles group knowledge workers with similar duties in the KiP. Capabilities are used to describe whether a knowledge workers has the right expertise to solve a specific task.	[DCMR13]
R23 Formalize interaction between knowledge workers	During the lifetime of a KiP knowledge workers might participate in several roles and interact with each other. This requires mechanisms that allow structured and unstructured protocols for the knowledge workers' communication and collaboration.	[DCMR13]
R24 Define knowledge workers' privileges	Privileges are an important requirement to protect data and knowledge elements from unauthorized access. Knowledge workers' privileges have to be defined explicitly to specify permissions for creating and modifying data elements.	[DCMR13]

Name	Description	Source
R25 Late knowledge workers' modeling	Due to the emergent characteristic of KiPs, roles and capabilities of knowledge workers have to be added or modified at runtime since they might acquire new skills.	[DCMR13]
R26 Late privileges modeling	Privileges associated to data and knowledge elements might have to be added or modified for knowledge workers during process enactment. Privileges have to be defined for knowledge entities that arise during the KiP.	[DCMR13]
R27 Capture knowledge workers' decisions	Decisions made by knowledge workers at runtime with their impact on the progression of the process have to be captured. Decisions can be related the selection between alternative execution paths or the manipulation of relevant data.	[DCMR13]
R28 Encouragement of knowledge workers	Knowledge workers have to be encouraged to maintain work plans and regularly update the status of their tasks. This is important since the work templates emerge through the individual adaptations of knowledge workers.	[MR14]

Table 2.6.: Name, description and source of knowledge workers' requirements for KiPs

KiPs depend on external events from the environment that influence the execution of the processes, e.g., the completion of a task or the modification of data from an external information system. In many scenarios knowledge workers use specialized tools for their work. These tools can be integrated with events to trigger the completion of tasks or the exchange of data. Another possibility is the integration with other case and workflow management tools. The following table contains the requirements that are related to the environment:

Name	Description	Source
R29 Capture and model external events	External events are triggered that influence the running processes by modifying values of the information model. Software support for KiPs needs to allow explicit events with their associations on the information model.	[DCMR13]
R30 External events late modeling	New external events have to be formalized by knowledge workers during the process enactment to associate new events.	[DCMR13]

Table 2.7.: Name, description and source of environment requirements for KiPs

2.2.2. Scenario A: Innovation Management

The ability to constantly develop innovations is important for an organization to generate growth and ensure sustainable success. Innovations describe qualitatively new products or processes that differ from what existed before in organizations [HS11]. Innovations need to be distinguished from ideas through their ability to be commercialized [Po11]. Existing literature provides various processes for innovation, whereas many of them are rooted in the stage-gate model that was introduced in 1990. The stage-gate model provides a conceptual and operational model to guide the launch of products from ideas [Co90]. The history of this model evolved over three generations up to today. The first generation has its roots in the phase project planing of the NASA. In this generation the development process only proceeds to the next phase after all tasks of the previous phase are completed. In the second generation the innovation process is divided into a predetermined set of stages with a gate at the beginning. Gates are used as quality control checkpoints before entering a stage with quantitative and qualitative measures. Decisions in these gates are usually made by multidisciplinary teams of senior managers that allocate required resources for the project. In the third generation the flexibility and speed of the innovation process is improved [Co94]. Activities in the latest version are not restricted to specific stages and the process is more adaptable. Stages in the third generation are overlapping and this allows to start with activities before or after the subsequent stage is entered. Decisions that are made in the gates might also be postponed if not enough information to make the decision is available, i.e., projects might continue in the other stages although the decision in one gate is not made. Gates and stages might also be skipped in case this is necessary for the project to complete.

Although every new generation of the stage-gate model improved the flexibility of the innovation process, early stages of the process including the idea generation are still an area of improvement for future work. According to the originator of the stage-gate model the seeds of success or failure are sown in the first steps of the process [Co90]. Considering the definition of collaborative knowledge work processes applied in this thesis, the early stages require a high degree of flexibility and creativity at runtime to solve complex problems. Agile methods are increasingly applied to allow for more adaptability in the early stages of the innovation process [Li14]. With this approach activities in the early stages can be repeated by iterations to manage the uncertainty. Although existing tools for innovation processes are able to support the general structure of the stage-gate model, they provide no means to support the early and creative stages of the innovation process. On the over hand there are approaches for open innovation that increase the number of participating stakeholders contributing or rating innovations. Improved process support for innovation management could enable creative solutions through open innovation with the structure of a third generation stage-gate model. We conducted a case study in one of the largest German software companies to investigate the idea generation process. This case study is described in detail in the master's thesis in [Ut14]. In this company two software solutions are already in place to support the idea generation. In the first solution (portal A) a highly structured process is used to assess ideas provided by employees in the organization. In the second solution (portal B) the process is entirely unstructured and allows the collaborative development of innovations. In portal A employees suggest incremental ideas that have to be well elaborated, while portal B can be used for disruptive innovations that have an impact on the business model. In the following, the case study based on these two portals is summarized.

Case Study 1: Idea generation
The software company uses two innovation management portals for the development of innovations through the internal workforce. One solution (portal A) is used for the submission and assessment of incremental ideals through reviewers. Most of the ideas in portal A are related to improvements of existing operational processes, e.g., making room reservations and catering for meetings and events. Incremental ideas are mainly concerned with the improvement of existing products or processes. Usually incremental ideas are proposed by individual employees who might receive a bonus in case their proposals are accepted by the reviewers, e.g., 10% of the cost savings caused by the proposal. The process for the development and assessment of incremental ideas is highly structured and predictable in portal A. The second innovation management portal (portal B) is much more unstructured and involves many employees in the development of disruptive innovations that might result in new products or processes. While innovations in portal A are usually well elaborated, innovations in portal B are in most cases in an early stage and other users can rate them. Portal B provides no process support for the collaborative development of ideas at the current stage. In the future the company needs to integrate both innovation management portals in one process because they want to combine the advantages of both approaches. This is necessary because the usage of two portals creates several problems in the company, e.g., ideas might be taken from portal A and placed in portal B.

The relevance of the requirements for KiPs in this application scenario is shown in Table 2.7. Regarding the *data requirements* this scenario requires access to shared data since employees can collaboratively refine ideas. All ideas are stored in the system even if they are not implemented or accepted. Thereby, employees can search for similar ideas that were submitted previously. The data model needs to be comprehensible because the system is rarely used by employees in the company. Highly relevant *knowledge action* requirements are related to the late actions modeling since the idea generation process requires much flexibility in the early stages. Late knowledge action roles are necessary due to the dynamic assignment of reviewers. Both requirements related to *rules and constraints* are highly relevant. They are necessary for the structured process that is used for the reviewing and assessment of ideas. Except for the recommendation requirement, all other *process requirements* have a medium relevance in this application scenario. Nevertheless, recommendations might become more relevant in the future if the support for early stages of the idea generation is improved.

All requirements related to *knowledge workers* are highly relevant for the idea generation. The modeling of knowledge worker skills is necessary to identify appropriate reviewers for ideas. Privileges for knowledge workers are highly relevant since some of the information provided for reviewers should not be visible for everyone. These privileges might have to be changed at runtime in case additional reviews are requested for an idea. Capturing decisions of knowledge workers is an important requirement in this scenario. Depending on the maturity of the idea that is submitted, decisions of knowledge workers might affect the flow of actions within the idea generation. For incremental ideas that are already well elaborated it is not necessary to perform tasks for the collaborative voting and refinement of ideas. Knowledge worker empowerment is highly relevant, because as many employees as possible have to contribute to these ideas.

2.2.3. Scenario B: Enterprise Architecture Management

Organizations face challenges related to growing importance of information technology (IT) and the increasing complexity of their IT-landscapes. To cope with these challenges, enterprise architecture (EA) covers business and IT aspects with their relationships in a model to provide a holistic view of an organization. EA models typically consist of infrastructure elements, business applications, business processes and relationships among them. Different viewpoints on this model are used to support decision makers with relevant information. EA management (EAM) is promoted as an approach to improve the alignment between business and IT, realize cost savings and increase failure tolerance [LW04, RWR06, Ro03]. Depending on the concerns of the decision makers EAM provides several instruments to manage the evolution of an EA, e.g., visualizations, architecture principles and processes. While documentation and analysis of EA models is supported in existing tools for EAM, support of processes for the management of the EA is missing to a large extent. This issue has been identified in a literature review on critical challenges in EAM by Lucke et al. in [LKL10]. According to the authors *"no formal steps exist for defining, maintaining and implementing EA and EA frameworks are not rigid enough in describing these steps"* [LKL10]. The process of managing an EA can be considered as unstructured or loosely structured with respect to the classification in the process management spectrum shown in Figure 2.2. Another challenging issue is that these processes have to be adapted to the organizational context and goals for every organization. In practice the structure of the processes often emerges based on the contributions of involved stakeholders. Due to these characteristics existing workflow management solution are not suitable to support processes in this context.

An EAM function usually comprises several different processes that would benefit from an improved software support. Examples for these processes are the development of a planned state, documentation of the EA, definition of transformation road maps, or standardization of technologies in the application landscape [Mo09, Bu08]. Depending on the maturity of the function in the organization, additional processes might have to be supported. One prominent example of an EAM process that can be found in literature is the Architecture Development Method (ADM) from TOGAF [Th09]. On the highest level this process consists of eight phases that can be structured in four iterations for the architecture context, architecture delivery, transition planing and architecture governance. For every phase the framework describes several tasks with the documents that have to be produced. This information provides valuable best practice knowledge for a work template that could be used as starting point to configure an organization-specific method. Nevertheless, this method cannot be directly used in an organization since usually not all tasks proposed by the framework have to be executed. The practitioner survey presented by Hauder et al. in [Ha14b] revealed that the majority of enterprise architects apply agile principles in their work to tackle this challenge. This requires a high degree of flexibility during the execution to perform the necessary adaptations. For this application scenario we conducted a case study in a German insurance organization. In the case study the development of a planned state of the EA based on the TOGAF ADM is evaluated. For this purpose the process is adapted to the context of the insurance organization. A detailed description of the case study including expert interviews with five stakeholders can be found in [Ha14a]. In the current stage this process is only roughly described on presentation slides and there is no software solution in place that can be used for this purpose.

Case Study 2: Development of a planned state
The development of a planned state of the EA consists of the three subsequent phases business architecture, information systems architecture and technology architecture according to the ADM of TOGAF. The subsequent phases of the ADM are not considered in this case study, because they are related to the implementation of the planned state. The framework provides tasks for every phase that have to be executed, whereas tasks might have logical dependencies among each other. For example the task refine and update versions of the architecture vision phase of the information systems architecture phase needs to elaborate the business drivers, goals, principles and the statement of architecture work documents. This task can only be started when certain previous tasks in the phase business architecture are completed. Although the framework provides many tasks, it is not possible to foresee all tasks with their required documents during the definition of the work template for the development of a planned state in practice. During the process new concerns of stakeholders might arise that have to be incorporated in the development of the planned state. Many stakeholders are usually involved in the development of a planned EA state, e.g., business architects, IT architects and members of the management board. In case responsible stakeholders are not available it is necessary to delegate or skip tasks.

The relevance of the requirements for this application scenario is illustrated in Table 2.8. All requirements related to *data* are highly relevant for this scenario since every task is related to the creation of documents or models. The system needs to provide data modeling capabilities to describe the EA model. This model might change during the development of the planned state, e.g., if it has to be extend with new concepts. For the creation of reports stakeholders require access to this data that is created during the process. The comprehensibility of the data model is important for users that are not familiar with EA. Requirements related to *knowledge actions* are highly relevant since the TOGAF ADM explicitly describes actions that are related to data. Late knowledge action roles could only be derived implicitly from the ADM. *Rules and constraints* are highly relevant due to the dependencies that are described for tasks between subsequent phases. *Goals* are explicitly modeled as part of the data model within the architecture vision phase. The most important *process requirements* are the support for different modeling styles and the flexible execution of processes that allows the handling of exceptions. Support for different modeling styles is necessary since the EA model can be described structured or unstructured [FHS13]. The learning from data is considered as highly relevant to allow the integration of existing information sources [Bu12, Ro13].

The most important *knowledge worker* requirements are the modeling of resources and the encouragement of stakeholders. The resources are necessary to model providers of relevant EA data, e.g., productive systems that contain EA data or stakeholders that have the knowledge. The maintenance of an EA is a collaborative effort that spans many departments in an organization. Encouraging knowledge workers to contribute with their knowledge is highly important for a successful initiative in this context. Finally, both *environment* requirements are highly relevant for the development of a planned state. External information sources can provide events that have an influence on the process, e.g., trigger about projects that started or new applications in the landscape. Results of an empirical evaluation of relevant information sources for EA that we conducted can be found in [Fa13].

2.2.4. Scenario C: Requirements Engineering

Requirements describe the functionality of a system in a way that is measurable and testable [IE05]. A precise understanding of requirements is crucially important to avoid expensive subsequent errors during the system design and implementation. Requirements engineering is concerned with discovering, developing, tracing, analyzing, qualifying, communicating and managing requirements that define a system [HJD10]. Existing methods for requirements engineering can be separated into activity-oriented and artefact-oriented approaches. In activity-oriented approaches the requirements engineering process can be divided into four main phases for elicitation, modeling, validation and verification [HL01]. During the elicitation phase needs of stakeholders and other resources are gathered. The modeling phase formally and informally structures the obtained requirements in a specification. In the final phase this specification is verified and validated on its correctness. Main problems of activity-oriented requirements engineering approaches are that they are difficult to customize and the syntactical quality of the resulting documents is not considered. Artefact-oriented approaches are focused on the resulting documents and models of the requirements engineering process to circumvent this issue [Fe10]. According to the definition applied in this thesis artefact-orientation can be considered as a collaborative knowledge work process since tasks are used to coordinate the creation of artifacts, data is integrated in the execution of the process and high flexibility at runtime is required. The approach suggests several reference models that are tailored for specific domains, e.g., business information systems. In addition, reference models can also be tailored during their execution by adding or removing artefacts from the model. We consider work templates as means to implement reference models for artefact-oriented requirements engineering in our solution.

At the current state the model for artefact-orientation is mainly used to structure and organize the outcomes of the requirements engineering activity, i.e., major concepts concerned with artefact-orientation. Software support for artefact-oriented requirements engineering is not available at the current stage to the best of our knowledge. The artefact model is divided into two parts for artefact structure and artefact content. In the artefact structure the composition of artefacts and content items is described. Artefacts are created by a sequence of tasks and they serve as a container for a certain number of content items. Content items are at the lowest level of the artefact hierarchy and represent the output of a single task [Fe10]. Content items can be described in different ways, e.g., a use case can be described textually or with an activity diagram and at different levels of abstraction through views. This artefact model is associated to the generic process model that is necessary to coordinate the creation of the artefacts. The generic process model consists of phases that are divided into activities at the highest level. Activities are performed by certain roles and divided into several tasks. Every task is concerned with the completion of artefacts or content items with input and output relationships. Sequences of tasks are described with use relationships and might be restricted through dependencies that are described on a lower level in the artefact model. Within a master's thesis the requirements for the software support of the artefact model is investigated in detail [Bi14]. In this thesis a case study for the elicitation of requirements based on a software for cash dispensers is conducted. In contrast to the previous case studies the elicitation of requirements is based on a fictive example without the involvement of an actual enterprise.

Case Study 3: Elicitation of requirements
The replacement of old cash dispensers through new machines in a bank requires the development of new software for the machines. Main motivation for the development of the new cash dispenser software is usability, maintainability and an improved overall quality. In this case study an artefact-oriented approach for the elicitation of the requirements is chosen. The approach is based on the reference model for business information systems analysis (BISA). During the requirements engineering for the new software a stakeholder model needs to be developed to describe who is working with the cash dispenser software. The stakeholder model is one part of the content items that are located in the context specification artefact. Within the context specification one of the tasks is related to the creation of the stakeholder model. The responsible requirements engineer that is working on this content item is responsible for the description of the involved stakeholders that are using the new cash dispenser software. This task is finished after the content item contains a detailed description for every stakeholder. Technically speaking the stakeholder model is described with a Unified Modeling Language (UML) diagram that is stored as file.

Table 2.8 illustrates the relevance of the requirements for the elicitation of requirements. The most relevant requirements for this application scenario are located in the category for *data* requirements. The artefact-oriented requirements engineering approach requires data modeling capabilities for the artefacts. Artefacts can be hierarchically organized with arbitrary content items on the lowest level. The approach explicitly mentions the requirement for late data modeling. Requirements engineers have to be able to access this data directly, i.e., to search for the stakeholder model. Data model comprehensibility is highly relevant because it describes the mandatory results that have to be produced. Among the requirements for *knowledge actions* two requirements are highly relevant. Data-driven actions are explicitly defined in the metamodel for artefact-orientation [Fe10], i.e., tasks are associated with artefacts. Due to the tailoring of reference models late actions modeling is highly relevant as well. *Rules and constraints* are explicitly mentioned with dependencies between artefacts. *Goals* are captured within an own artefact in the context specification.

Only three requirements related to *processes* are explicitly mentioned in the artefact model. Different modeling styles are important due to the variety of diverse models that are used during the elicitation of requirements, e.g., activity diagrams, word documents and use case diagrams. The flexible execution is mentioned as one of the key requirements that delineates artefact-oriented from activity-oriented approaches. Artefacts can be created with specific requirements engineering tools, i.e., learning from data within these tools is another highly relevant requirement. Only one explicit requirement related to *knowledge workers* could be identified in the case study. Decisions of requirements engineers have to be captured for the tailoring of the reference model, e.g., whether some artefacts are removed or tasks are skipped. All other requirements in this category are rated with a medium relevance even though they are not explicitly mentioned, because they are important for the software support in our opinion. Similar to the case study for the development of a planned EA state, the elicitation of requirements needs to integrate events from the environment. These events are used to integrate specialized requirements engineering tools, e.g., to capture the state of artefacts created in other systems.

Requirement	**Relevance**		
	Scenario A	**Scenario B**	**Scenario C**
Data			
R1 Data modeling	◐	●	●
R2 Late data modeling	◐	●	●
R3 Access to data	◐	●	●
R4 Access to shared data	●	●	●
R5 Date model comprehensibility	●	●	●
Knowledge Actions			
R6 Data-driven actions	◐	●	●
R7 Late actions modeling	●	●	●
R8 Knowledge action roles	●	●	◐
R9 Late knowledge action roles	◐	◐	◐
Rules and Constraints			
R10 Rules and constraints	●	●	●
R11 Late constraints	●	●	●
Goals			
R12 Goal modeling	●	●	●
R13 Late goal modeling	◐	●	◐
Processes			
R14 Different modeling styles	◐	●	●
R15 Visibility of knowledge	◐	◐	◐
R16 Flexible execution	◐	●	●
R17 Unanticipated exceptions	◐	●	◐
R18 Migration of instances	◐	◐	◐
R19 Learning from event logs	◐	◐	◐
R20 Learning from data	◐	●	●
R21 Recommendations	○	◐	◐
Knowledge Workers			
R22 Resource/skill modeling	●	●	◐
R23 Workers' interaction	●	◐	◐
R24 Workers' privileges	●	◐	◐
R25 Late resource modeling	●	◐	◐
R26 Late privileges modeling	●	◐	◐
R27 Workers' decisions	●	◐	●
R28 Knowledge worker encouragement	●	●	◐
Environment			
R29 Model external events	○	●	●
R30 Late modeling of events	○	●	◐

Description for icons:

● High relevance
◐ Medium relevance
○ Basic relevance

Table 2.8.: Relevance of the generic KiP requirements for the three application scenarios

2.3. Related Work

In this section related approaches for process support of knowledge work are described based on the results of an extensive literature review. Figure 2.7 characterizes the applied literature review in this thesis based on the framework presented in [Fe06]. The type of literature review is natural language and we are focused on theories as well as experiences with process support for knowledge work. Theories are mainly related to process management approaches that are suitable to support enough flexibility at runtime for KiPs according to current business process management (BPM) literature. Next to the current state of theoretical approaches in BPM literature, the related work section covers experiences on case management use cases. We cover the entire content that is available and explicitly refers to the topic of the thesis. The perspective of the literature review is neutral since all approaches are treated equally. Our goal is to comprehend the entire historical development of the field from the perspective of practitioners as well as researchers. Due to the high practical relevance of the field, literature from practitioners provides valuable case studies from various different domains, e.g. healthcare, insurance, court case management and public sector. In the literature review we explicitly look for future research directions to identify common issues regarding process support for knowledge work.

Characteristic		**Category**			
Type		*natural language*		mathematical-statistical	
Focus		research results	research method	*theory*	*experience*
Target	Formulation	not explicit		*explicit*	
	Content	*integration*	*criticism*		*central topics*
Perspective		*neutral*		position	
Literature	Selection	not explicit		*explicit*	
	Extensiveness	foundations	representative	selective	*complete*
Structure		*historical*	thematically		methodical
Target Group		*common public*	practicioners	common researcher	specialized researcher
Future Research		not explicit		*explicit*	

Figure 2.7.: Characterization of the literature review according to Fettke et al. [Fe06]

Relevant articles are searched from leading journals, conferences, workshops as well as book chapters with IEEExplore, CiteSeerX, Google, Google Scholar and the library of our research institution. Due to the low maturity of the research area only few publications are available as journal publications at the current stage. The scope is narrowed to publications in the information systems (IS), business process management (BPM) and computer-supported cooperative work (CSCW) fields since these are most relevant for the research area. We followed the approach presented by Webster et al. in [WW02] and used a variety of different keywords, e.g., knowledge-intensive process, case management, case handling, dynamic case management, adaptive case management, to detect relevant literature. Figure 2.8 illustrates the result of the literature review with a chronological overview of the most relevant related works with respect to this thesis until today.

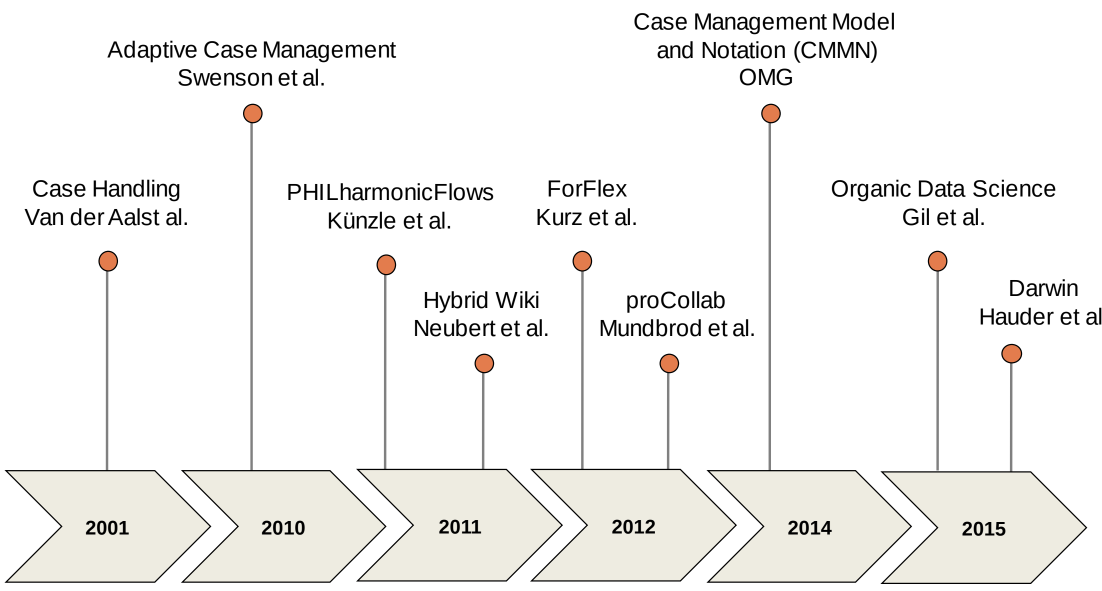


Figure 2.8.: Related work relevant for this thesis by publication year until today

In 2001, case handling has been initially proposed as an approach that goes beyond workflow management to support highly flexible processes. Initial publications related to case handling covered processes that are completely predefined at design-time and provided no solutions for adaptations at runtime. Based on case handling the notion of adaptive case management (ACM) has been presented in 2010. 41 publications were available on ACM by the end of June 2014, whereas the majority was published in 2012 or later [HPM14]. Main goal of ACM is to make the adaptations of cases possible at runtime through end-users. Hybrid Wikis were developed in 2011 as an approach to collaboratively structure information at the sebis chair. This project can be considered as a predecessor of this thesis since our work builds on the experiences with Hybrid Wikis. In 2012 the forFlex project provided an early prototype that is based on ACM design principles. At the same time the project proCollab presented a lifecycle for collaborative knowledge work. In 2014, the Object Management Group (OMG) released the final version of the Case Management Model and Notation (CMMN) as a standard for the specification of declarative process models. In the same year as this thesis is published the Organic Data Science approach has been presented to support open scientific processes. In the following section we will describe how this thesis extends related work in detail.

2.3.1. Case Handling

Case handling has been presented as a new paradigm to support flexible and knowledge-intensive processes. It was first introduced in 2001 by van der Aalst et al. [AB01]. In this approach processes are represented with cases as the central concept. In contrast to workflow management these cases are not solely driven by the control flow, but data is integrated in the execution of the processes. Activities in cases are also less rigid and knowledge workers do have more control. An important ability of case handling is the role concept for activities with execute, redo and skip compared to workflow management that only allows for an execution role. According to van der Aalst [AWG05] the four main features of case handling are:

1. Context tunneling is avoided since knowledge workers can access the entire data of a case. In workflow management solutions users usually only have access to data that is relevant for the particular task. Searching for data objects at any time is not possible in workflow management solutions.
2. Case handling integrates data in the execution of the processes. Activities in the case are enabled based on the availability of information objects. In workflow management solutions the course of activities is driven by the control flow. Basically the case handling system should not focus what *should* be done, but rather on what *can* be done in the case.
3. In case handling the distribution of work is separated from authorization. Activities can be delegated to other knowledge workers during the case if necessary. Only workers with the proper roles are allowed to distribute work with the delegate function. In a workflow management approach the delegation of activities is not possible and only authorized workers can complete activities.
4. Knowledge workers can access the information objects in a case independent from the activities given that they have proper access rights. They are allowed to search, view, add and modify information objects without an activity attached at any time. This provides the knowledge workers with much more freedom since they can comprehend the current status of relevant information objects.

Reijers et al. [RRA03] describe case handling with three main characteristics: (1) the system is focused on the case as central element, (2) the process is data-driven and (3) some parts of the process model are defined implicitly. The case covers all relevant concepts to structure the process with information objects, activities, forms and actors that might be assigned to several groups. Cases can also be organized in hierarchies to break down complex cases into subcomponents. Workflow management systems specify permitted steps with the control flow of a process *explicitly*. In case handling the definition of the process is less rigid and describes possible execution steps in a declarative way, i.e., forbidden traces are excluded while leaving maximum flexibility for remaining traces. As a result possible flows in the case are defined *implicitly*. After these initial publications on case handling several related definitions have been introduced, e.g., dynamic case management, emergent case management, production case management and adaptive case management (ACM). Section 2.3.2 summarizes existing definitions related to case handling and distinguishes them from ACM based on the characteristics of KiPs. A discussion on how flexible processes can be supported with both case handling and adaptive process management is provided by [GRA08].

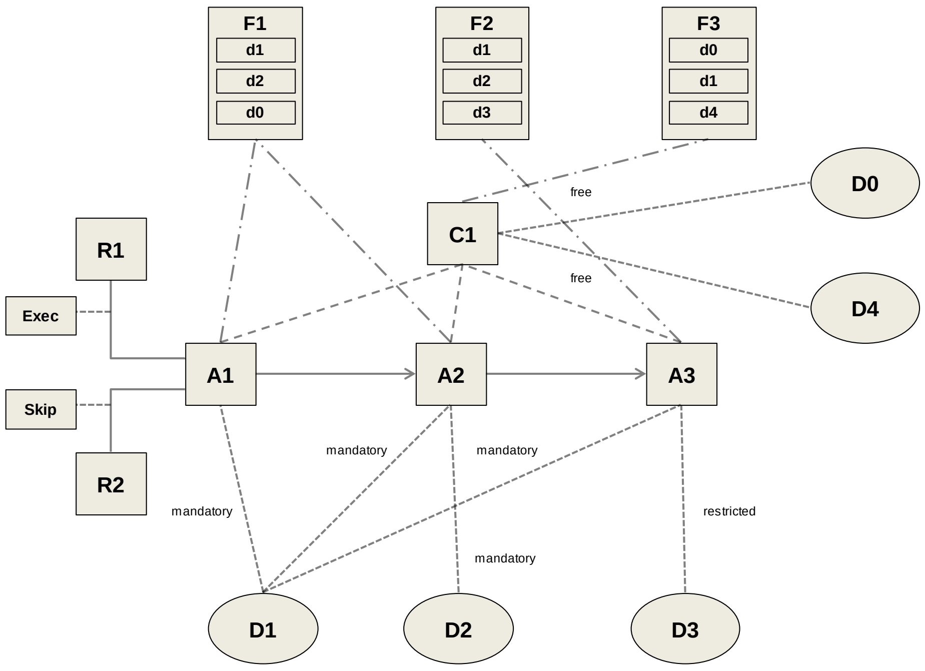


Figure 2.9.: Overview of the initial schema level metamodel for case handling [AWG05]

An example illustrating the case management metamodel is shown in Figure 2.9 that is taken from [AWG05]. The example shows a case with its data objects, forms, activities and roles. *C1* is a case definition that consists of activity definitions *A1*, *A2* and *A3*. Forms are used to provide different views on data objects in a case. Dotted lines in the figure represent the association between activity definitions and data object definitions. Forms are linked to activities to represented the most relevant data objects. Free data objects are linked directly to the case and can be changed at any time. Mandatory and restricted data objects are always linked to activities. As shown in the figure *D1* is a mandatory data object for A1, A2 and A3, i.e., this data object has to be entered to complete the activities. The form *F1* holds an entry for D1 since it is mandatory for this activity. The knowledge worker could already provide the data object *D2* in form F1 in case this information is already available. This would automatically finish the activity A2 as soon as A1 is completed as well.

This is not possible for *D3* since this data object is restricted to A3, i.e., it has to be completed in the associated activity and cannot be preferred. *D0* and *D4* are free data objects assigned to the case definition, i.e., knowledge worker may use the information to work on the case. Two roles are defined in the case that specify who is allowed to execute and skip activities. Knowledge workers assigned to *R1* are allowed to execute A1 and knowledge workers assigned to *R2* are able to skip A1. The additional roles provide a powerful mechanism to model a wide range of exceptions in case management. The redo role can be used to represent a loop if already completed tasks have to be repeated again. Skip allows the handling of exceptions that would otherwise have to be explicitly modeled in the case. In addition to these roles van der Aalst recommended additional roles that could be used in case management [AWG05] , e.g., responsible role for activities or case manager role for one case.

2.3.2. Adaptive Case Management

Various approaches related to case handling have been proposed after the initial publication in 2001. Within an extensive literature study we compared existing case handling definitions and investigated the rise of the term case management in [MHM15]. Three different types of definitions for approaches related to case management are extracted. First, the definitions are explicitly described in the publications. Second, the paragraph describing case management is used in case no formal definition is available. Third, individual sentences from the paper are extracted if the first two types are not applicable. In this study the definitions are compared with characteristics of KiPs presented by Di Ciccio et al. [DCMR13]. The detailed description of the comparison is publicly available[1] and omitted in this thesis for the sake of brevity.

Table 2.9 summarizes the results with the initial publication source. First references to the term case management were introduced in 1994 by Davenport and Nohria [DN94]. In this work case management is considered as valuable to empower workers that are involved in processes that deal with customers. In 2006 Kaan et al. [KRM06] tried to introduce case management as an alternative to case handling, whereas this attempt is odd compared to existing definitions. In subsequent definitions the term case management evolved with a stronger emphasis on its collaborative nature and the flexible interaction between humans, content and processes [MHM15]. The term ACM has been popularized as case management approach that allows end-users to create case definitions at runtime [Sw10a].

Definition	**Date**	**Characteristics of KiPs**							
		C1	C2	C3	C4	C5	C6	C7	C8
Case management role [DN94]	1994	○	○						
Case handling [AWG05]	2005	●		○	●	●			
Case management [KRM06]	2006			–	–		–		–
Case management work [Ke08]	2008	○	●	○	●				○
Dynamic case management [CMV09]	2009	○	●				●	○	
Case management [Wh09]	2009	●	●	○	○	●	●		○
Case management [Mc10]	2010	●	●	●	●	●	●		○
Case management [dPv10]	2010					●			
Case management [Sw10a]	2010								
Adaptive case management [Sw10b]	2010		○						
Adaptive case management [Pa10]	2010	○							
Emerging case management [Bö11]	2011	○	●	○					
Production case management [MNS13]	2013	●		●	○	○			
Adaptive case management [MNS13]	2013	●		●	●	○			○

Description for icons:
● Explicit in the definition
○ Implicit in the definition
– Precluded by the definition

Table 2.9.: Case management approaches compared with characteristics of KiPs [MHM15]

[1] `http://arxiv.org/abs/1507.04004`, last accessed on: 2015-07-20

Main difference of ACM to the previous case management approaches is the focus on end-users that define cases during the execution of processes without the involvement of software developers [MNS13]. As previously described for case handling, data is also a central aspect in ACM and has to be considered equally important. In [Sw10a], processes are classified with their degree of system interaction during the structuring of the processes ranging from designed to emergent. Structured processes are designed in a top-down approach by software developers or modeling experts. The focus of these processes is the reduction of costs and efficiency improvements through industrialization. At the other end of the work spectrum processes emerge bottom-up through adaptations of knowledge workers. ACM is classified as emergent and slightly more structured than ad-hoc social collaboration. These processes are focused on high quality of the resulting artifacts. ACM is considered as foundation for learning organizations that constantly have to adapt their processes [Sw13]. Table 2.9 indicates that there is little consensus about a common definition on case management, since none of the definitions covers all characteristics of KiPs. As a consequence we decided to propose a definition based on the characteristics of KiPs: *"Case management is a practice for knowledge-intensive processes with a case folder as central repository, whereas the course of action for the fulfillment of goals is highly uncertain and the execution gradually emerges according to the available knowledge base and expertise of knowledge workers."* [MHM15].

ACM is gaining interest by researchers and practitioners over the past five years since half of the publications related to this topic are published in this timeframe [HPM14]. Despite this increasing interest there are currently no solutions for ACM available that address the requirements presented in Section 2.2.1. The vast majority of literature available in this area is problem-oriented only describing important aspects an ACM system needs to fulfill or differences to workflow management solutions. Additionally, many case studies with applications in different application scenario are published in this community. In 2014, we conducted a database-driven literature review to identify research challenges in ACM [HPM14]. In this study we identified 77 codes referring to research challenges in ACM. Every code is one instance of a challenge that could be found in a scientific publication or book chapter. These codes we grouped to 13 research challenges and categorized in five distinct areas for data integration, knowledge worker empowerment, theoretical foundation, authorization and role management, as well as knowledge storage and extraction. As of June 2014, we were able to collect 32 publications and book chapters with ACM research challenges. 16 of these publications were published in 2012 or later. The five research categories for ACM are presented in the following with examples taken from [HPM14]:

1. **Data integration:** The integration of data in the execution of the processes leads to three challenges that could be identified in literature. The *data storage* is challenging due to the large volume of data that need to be managed. While workflow management systems only manage data related the process structure, case management systems need to store everything related to the case similar to a content management system (CMS). Data related to the case requires version management to restore previous states. *Shared memory* is necessary to provide enough context information for the completion of tasks, which is related to the problem of context tunneling in case handling. During the concurrent editing of data in the cases conflicts might arise when a file is changed by more than one person. Conflicts and inconsistent states of information objects have to be avoided through *data locking* mechanisms.

2. **Knowledge worker empowerment:** Empowering knowledge workers to define cases at runtime is an important issue in ACM and two research challenges could be identified in literature. *Advanced collaboration* deals with the allocation of work based on skills, competencies and availability of knowledge workers in the system. This requires a representation of the available resources in the system. In many situations case templates also have to be developed collaboratively to solve complex problems. The research challenge on *guidance techniques* contain codes related to the creation of templates for cases and intelligent user assistance through recommendations.

3. **Authorization and role management:** *Authorization* is concerned with access control challenges that avoid unauthorized access and manipulation of data. Important issues in the future could be digital signatures and dynamic access control to introduce new knowledge workers to the case. *Role management* refers to the more sophisticated role concepts that are necessary in case management, e.g., skip, redo and delegate roles. The assignment of roles might also have to be changed at runtime with new workers that join the case. Similar to the flexibility requirements for tasks and information objects, the roles for these concepts have to be dynamically created.

4. **Theoretical foundation:** Many research papers mention challenges that are related to the theoretical foundation of case management. A *common theory* or model for ACM is still missing in literature. One challenging issue is the *adaptation* of case templates that are performed at runtime since they might originate inconsistencies on already instantiated cases of this template. *Routine aspects* have to be integrated in ACM with a link to more structured process models like BPMN. Logical dependencies between tasks in the case and the link from information objects to tasks require the definition of *rules and constraints*.

5. **Knowledge storage and extraction:** The *storage of knowledge* is challenging due to the wide range of possible information object types that require advanced content management capabilities, e.g., searching, versioning and retention management. Compared to a workflow management solution, experiments revealed that the effort to define a case is much larger because many exceptions need to be considered. *Extracting knowledge* from the case definitions is necessary to ensure efficient use of the solution through knowledge workers. Currently, the usability of existing case management solutions is an issue and requires much more efforts in the future.

Due to the low maturity of the research area there are only few solutions for ACM available. One approach that builds on the presented ACM design principles is developed within the ForFlex project that is explained in Section 2.3.5. To the best of our knowledge there are no solutions that address the challenge of knowledge worker empowerment in current literature. Main goal of this thesis is to empower end-users to structure KiPs on wiki pages, i.e., wiki pages are extended with lightweight structuring concepts so that they represent cases. The foundation of our approach to support the collaborative structuring of KiPs is based on Hybrid Wikis that are explained in Section 2.3.4. On the one hand our solution can be considered as an ACM system since it allows end-users to define cases at runtime, but on the other hand modeling experts are also able to maintain templates for cases. Due to the strong emphasis on end-users current literature on ACM is not considering the role of modeling experts that are responsible for the maintenance of templates.

2.3.3. PHILharmonicFlows

In existing business applications the process-oriented view is separated from the data-oriented view. Main goal of this research project is the integration of data as driver for process specification and enactment, i.e., the progress of processes needs to be aligned with available object instances and their attribute values at runtime [KWR11]. These attribute values are summarized on form-based activities that can be invoked by authorized users. The authors of this project introduce the term *object-awareness* to define processes that address this issue. Figure 2.10 illustrates the data and process structure in object-aware processes. Data objects are comprised of an object type as well as a set of attributes, e.g., the object type job has two attributes for name and valid. Object types are associated to other object types through relations with cardinalities, e.g., jobs are associated with *0* to *n* applications. At runtime objects are instantiated with values for attributes and relations to other object instances. The process structure describes the behavior of an object and the interactions to other objects.

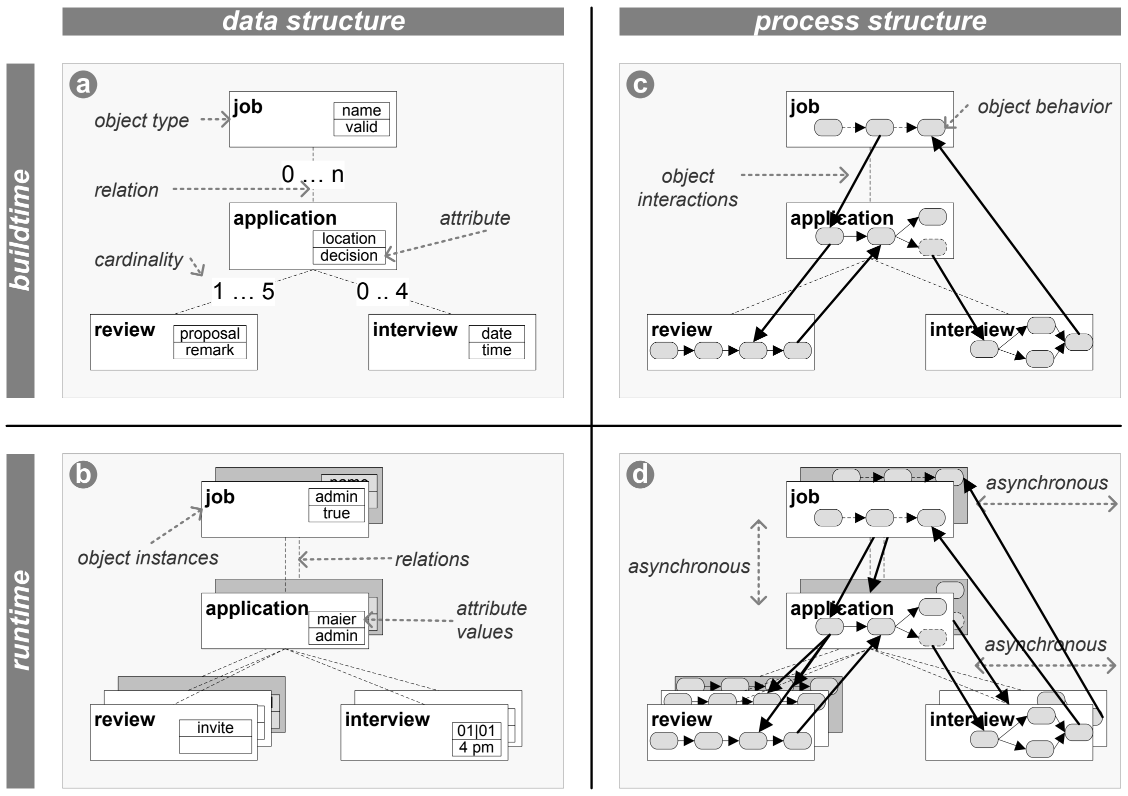


Figure 2.10.: Overview about data and process structure in object-aware processes [Kü13]

The state of an object instance is determined at runtime by the values of the attributes in the data structure. Attributes are grouped on activity-based forms for users and can be separated into optional and mandatory activities. The latter activities have to be completed to continue with the subsequent state in the object behavior. Figure 2.11 shows optional and mandatory activities for the review object of an application. The attribute values for urgency and return date have to be entered within a mandatory activity by the personnel officer. After the form for this activity is saved, the review continues from the state initiate to fill in. In the initiate state an employee might already provide information about the application in an optional activity. At the latest this activity has to be completed during the fill in state. Some values in the forms are greyed out since they cannot be edited by the assigned role. The framework also considers the modeling of access rights necessary to specify who is allowed to see and edit forms for the object instances.

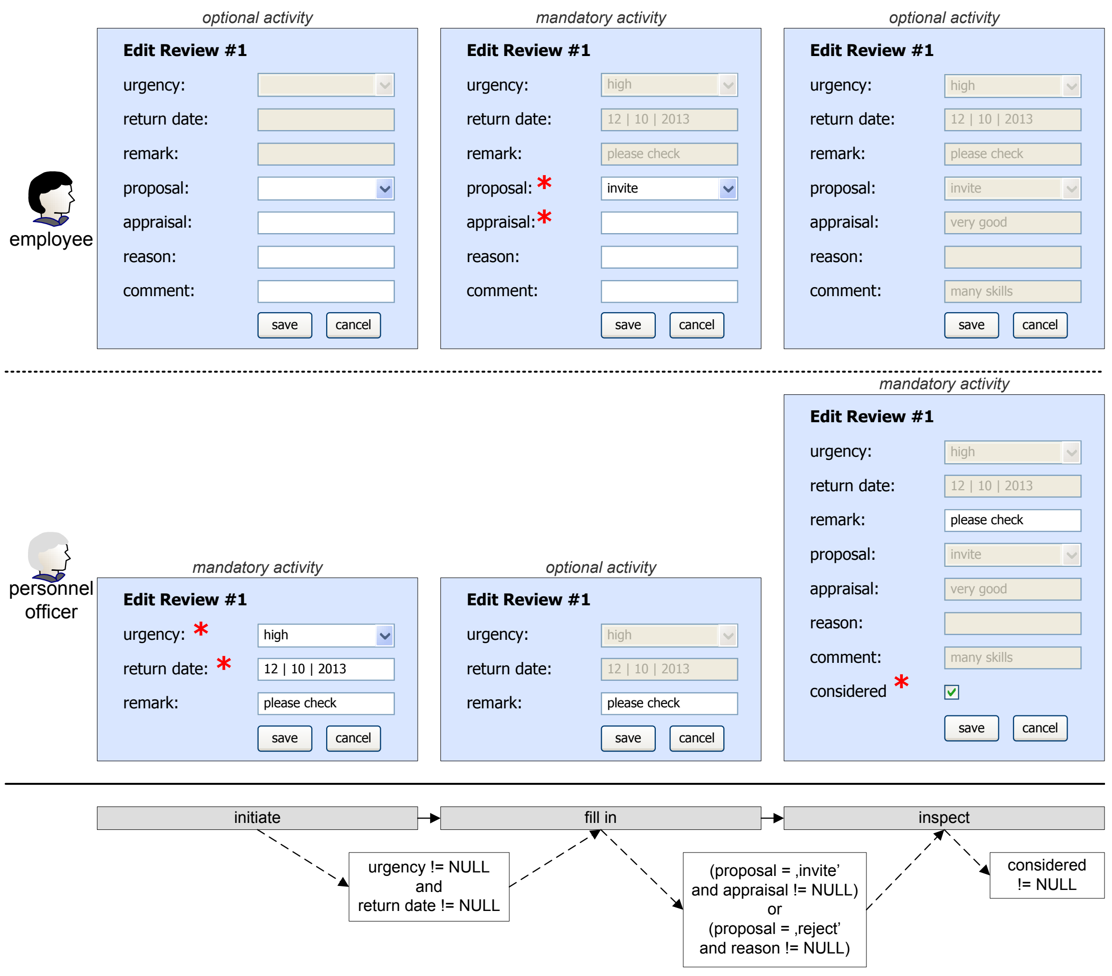
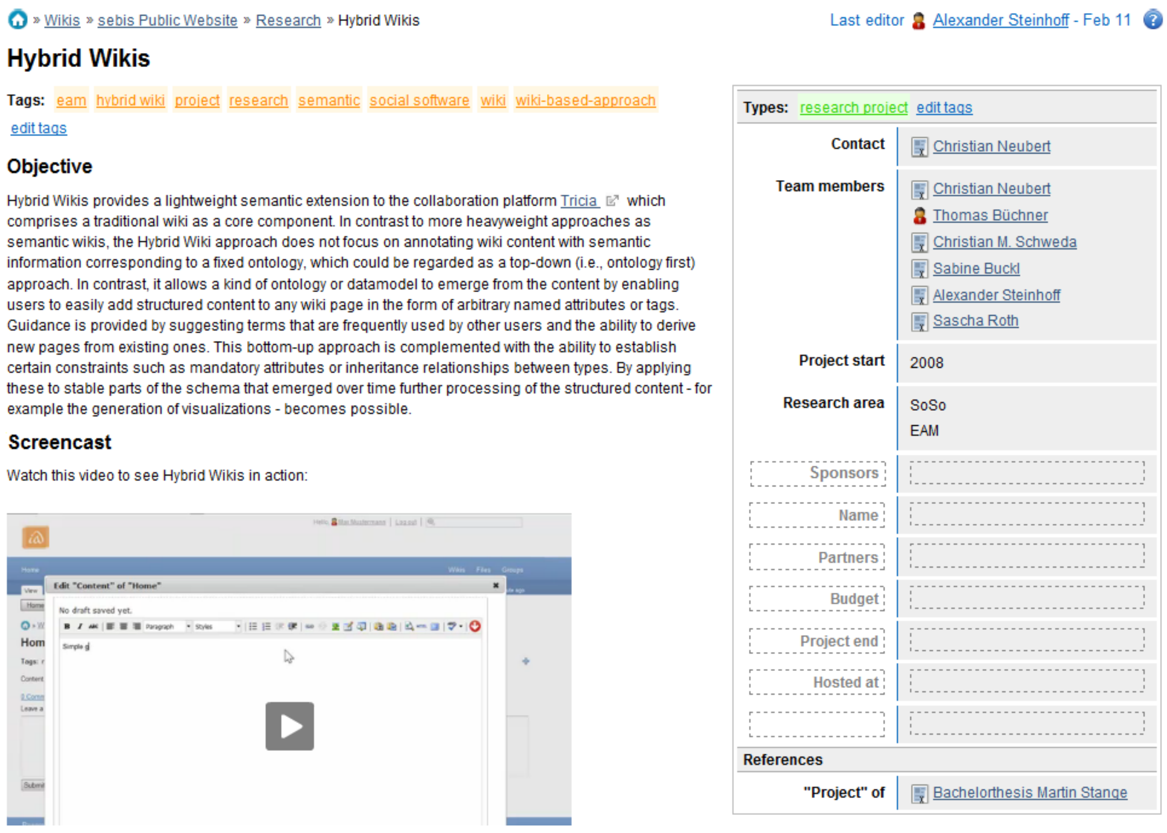


Figure 2.11.: Review state determined by attribute values of mandatory activities [Kü13]

Regarding its expressiveness the framework seems to be a promising approach to integrate data and processes in one approach, which is an important requirement for the software support of KiPs. Nevertheless, it is not evaluated to which extent knowledge workers can be empowered to model these object-aware process. Specifying the object and process structure requires advanced expertise in various modeling techniques, i.e., the barrier for knowledge workers might be even higher than in activity-centric approaches. In this thesis, we aim at empowering knowledge workers without expertise in all these modeling techniques to structure KiPs. Thereby, we provide lightweight structuring concepts as metaphors for knowledge workers to collaboratively structure the data and process aspects of KiPs.

2.3.4. Hybrid Wiki

Hybrid Wikis were developed in the PhD thesis of Neubert [MNS11] and used as productive system at sebis since 2009. Primary goal of Hybrid Wikis is to lower the barriers for non-expert users that need to structure information. For this purpose neither special wiki syntax or knowledge about modeling concepts should is necessary. Hybrid Wikis achieve this by using lightweight structuring concepts and metaphors that end-users are familiar with. The system can exploit the structure provided by end-users to query and organize the content, e.g., to query wiki pages that describe research projects. Hybrid Wikis have been successfully applied in various different contexts, e.g., Wiki4EAM [MN11], SmartNets [Ha13b] and as productive system at the sebis chair[2]. The term "hybrid" stems from the fact that only a subset of the features from semantic wikis are integrated in a classical wiki system. The structural concepts introduced in this subset are orthogonal and not limited to one application domain. Figure 2.12 shows a screenshot of a Hybrid Wiki page that describes this particular project.



» Wikis » sebis Public Website » Research » Hybrid Wikis

Last editor Alexander Steinhoff - Feb 11

Hybrid Wikis

Tags: eam hybrid wiki project research semantic social software wiki wiki-based-approach
edit tags

Objective

Hybrid Wikis provides a lightweight semantic extension to the collaboration platform Tricia which comprises a traditional wiki as a core component. In contrast to more heavyweight approaches as semantic wikis, the Hybrid Wiki approach does not focus on annotating wiki content with semantic information corresponding to a fixed ontology, which could be regarded as a top-down (i.e., ontology first) approach. In contrast, it allows a kind of ontology or datamodel to emerge from the content by enabling users to easily add structured content to any wiki page in the form of arbitrary named attributes or tags. Guidance is provided by suggesting terms that are frequently used by other users and the ability to derive new pages from existing ones. This bottom-up approach is complemented with the ability to establish certain constraints such as mandatory attributes or inheritance relationships between types. By applying these to stable parts of the schema that emerged over time further processing of the structured content - for example the generation of visualizations - becomes possible.

Screencast

Watch this video to see Hybrid Wikis in action:

Types: research project edit tags

Contact	Christian Neubert
Team members	Christian Neubert Thomas Büchner Christian M. Schweda Sabine Buckl Alexander Steinhoff Sascha Roth
Project start	2008
Research area	SoSo EAM
Sponsors	
Name	
Partners	
Budget	
Project end	
Hosted at	
References	
"Project" of	Bachelorthesis Martin Stange

Figure 2.12.: Screenshot of a Hybrid Wiki page showing the structured attributes [MNS11]

Every wiki page can be organized with sub pages and consists of unstructured content (rich text) and structured content (types and attributes). The example shows a wiki page of type research project with several predefined attributes for contact, team members, project start and research area. Additionally the system proposes several optional attributes that are used in related wiki pages. Pages can be bi-directly associated through references, e.g., the Hybrid

[2]https://wwwmatthes.in.tum.de/, last accessed on: 2015-05-15

Wiki page is the project of the bachelor thesis. The referenced pages in the attributes also have a defined type, e.g., team member and student project. New attributes can be directly added on the page by clicking on the dashed boxes. Hybrid Wiki types and attributes are described in the following:

- **Attributes:** Basically attributes are key-value pairs that are attached to wiki pages. It is possible to assign multiple values to one attribute. Every attribute value needs to have a data type assigned, e.g., date, string, or integer. In addition, simple constraints can be specified for attributes to ensure that the value is not empty or fulfills some syntactical requirements. In order to encourage end-users to structure the wiki page, recommendations for attributes are displayed at the bottom of the attribute box. A new attribute is created as soon as the end-user enters a value for one of the recommended attribute. Furthermore, suggestions for attribute names and values are shown when the users starts typing. This improves the consistent usage of terms in the Hybrid Wiki.
- **Types:** Types are used to make a statement about the wiki page. In the initial version of the Hybrid Wiki, end-users were allowed to provide an arbitrary number of types for a wiki page. Subsequent releases limited the number of types to exactly one since this proved to produce more reliable results. Main purpose of the types is to determine the set of attributes that are shown on the page. Types are used to generate lists of pages of the same type. By clicking on a specific type in the user interface, all wiki pages that have this type assigned are shown in a list.

This approach provides valuable and proven solutions with respect to some of the requirements for software support of KiPs presented in Section 2.2.1. In particular four of the five data requirements are already decently fulfilled by Hybrid Wikis. They allow the (late) data modeling through end-users that can add optional attributes to wiki pages at runtime, which is an important requirement for KiPs. Furthermore, the approach has proven to advance the data model comprehensibility for end-users through its lightweight structuring concepts and metaphors. They accept limited modeling capabilities of end-users and suppose no previous experience with specific markup languages. Hybrid Wikis already provide means for the learning from event logs for attributes, which is another requirement mentioned in Section 2.2.1. These capabilities of the Hybrid Wiki are also directly related to the data integration challenge that has been revealed through the literature review on ACM in [HPM14].

Although Hybrid Wikis are promising for the collaborative structuring of content in a variety of different application scenarios, they are not suitable to support processes of knowledge workers at the current stage. One of the limitations of the Hybrid Wiki is missing access rights for attributes since users can only be authorized for entire pages. Attributes in KiPs require fine-grained authorization concepts to specify who is allowed to edit and read data on a wiki page. Another limitation regarding process support for knowledge work is the missing notion of tasks in the Hybrid Wiki. It is not possible to describe any behavioral elements of types except for simple status attributes. The solution presented in this thesis builds on the experiences made with Hybrid Wikis and extends them with new structuring concepts that are inevitable to support KiPs. Although wikis are increasingly used in organizations there is no solution available that is able to support KiPs in a wiki to the best of our knowledge.

2.3.5. Service-Oriented IT-Systems for Highly Flexible Business Processes

Within the forFlex project the authors present a solution that is based on ACM to support loosely structured KiPs [KH12]. This approach is based on initial publications on ACM by Swenson et al. [Sw10a] and the authors present a process model with a prototypical implementation of their approach. Their process model consists of three subsequent phases that are repeated in iterations with *execution*, *control* and *overarching case adaptations*. During the execution a template for the given case is instantiated. Before and during the execution of the cases the template is adapted to the specific needs. According to the authors this *case specific adaptation* can be compared to agile software development approaches. Smaller adaptations can be directly incorporated in the case, while more complex innovations require the approval of a case manager. As soon as the goals are achieved, the case can be closed and the subsequent control phase starts. During the control phase the completed case is assessed based on its efficiency and goal attainment. After the assessment is completed the final phase starts with the overarching case adaptation. During this phase valuable improvements that might be beneficial for other cases are selected and generalized in the case templates. The authors present eleven different roles that are assigned to these phases with responsibilities. Figure 2.13 illustrates the design of the ACM approach in the forFlex project.

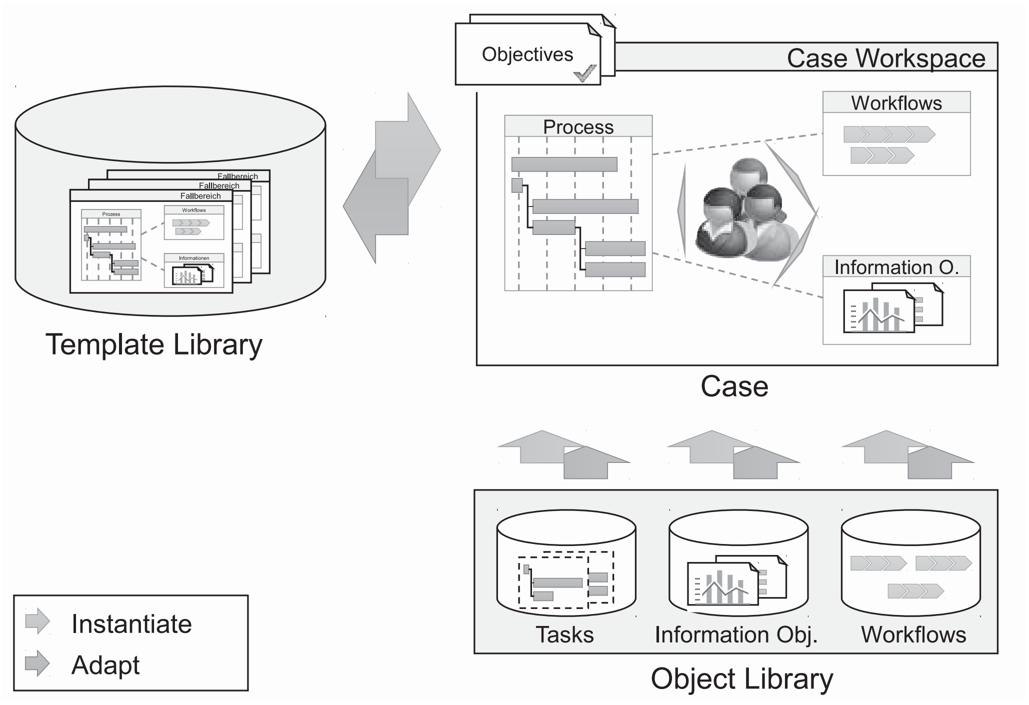
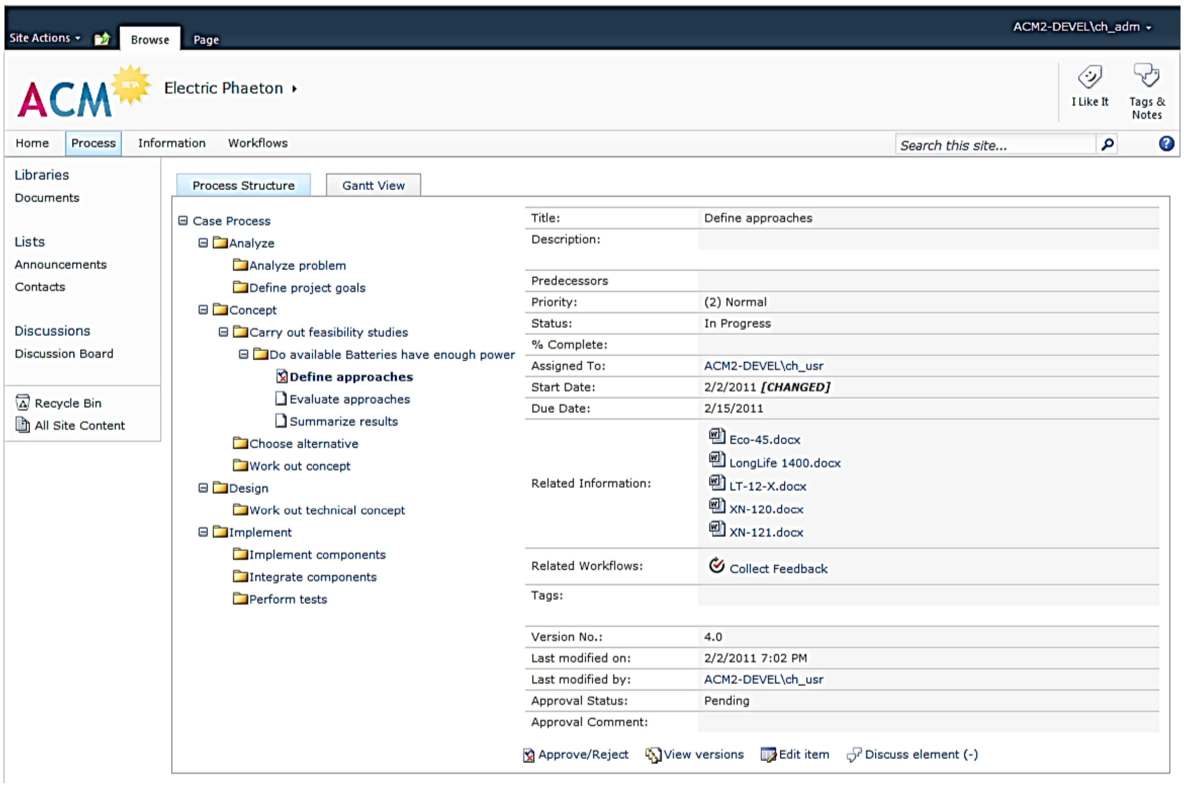


Figure 2.13.: Design for an ACM system that supports KiPs according to Kurz et al. [Ku15]

The template library consists of generalized best practices that are maintained during the overarching case adaptation phase. Instantiations of these templates are called cases in this approach. Every case consists of objectives that describe the goal(s) that should be achieved after the completion of the case. Cases consist of tasks, workflows and information object elements. During the case specific adaptation new elements can be instantiated in the case from an object library. Figure 2.14 illustrates a screenshot of the prototype that implements this approach based on Microsoft Sharepoint[3]. Tasks can be structured in a hierarchical tree with a name, metadata for start and end date as well as attached documents.



- Case Process
  - Analyze
    - Analyze problem
    - Define project goals
  - Concept
    - Carry out feasibility studies
      - Do available Batteries have enough power
        - **Define approaches**
        - Evaluate approaches
        - Summarize results
    - Choose alternative
    - Work out concept
  - Design
    - Work out technical concept
  - Implement
    - Implement components
    - Integrate components
    - Perform tests

Title:	Define approaches
Description:	
Predecessors	
Priority:	(2) Normal
Status:	In Progress
% Complete:	
Assigned To:	ACM2-DEVEL\ch_usr
Start Date:	2/2/2011 ***[CHANGED]***
Due Date:	2/15/2011
Related Information:	Eco-45.docx LongLife 1400.docx LT-12-X.docx XN-120.docx XN-121.docx
Related Workflows:	Collect Feedback
Tags:	
Version No.:	4.0
Last modified on:	2/2/2011 7:02 PM
Last modified by:	ACM2-DEVEL\ch_usr
Approval Status:	Pending
Approval Comment:	

Figure 2.14.: Screenshot of the ACM prototype implemented in Microsoft Sharepoint [KH12]

Although the approach is a sound implementation of the ACM approach, several requirements for software support for KiPs are not considered in this approach (cf. requirements for KiPs described in Section 2.2.1). It is not possible to describe logical dependencies between tasks since they can only be hierarchically organized in the solution. The authors already identified CMMN as one possibility to specify dependencies in the processes [Ku15]. Nevertheless, CMMN is currently not integrated in the latest version of the prototype. Another limitation is the simplified representation of information objects since only documents can be considered and it is not possible to specify information types. The authors also mentioned that the missing information structure makes it very challenging to find already existing entities [Ku15]. This might be an obstacle for the continuous refinement of templates, because workers might not be able to reuse already existing templates. Roles and authorization mechanisms are missing as well in the prototype, e.g., it is not feasible to delegate or execute tasks. The documents are not integrated in the execution of the process, i.e., the progress is not automatically computed based on the documents associated to a task. In the same publication, the authors also mentioned that incentives in the prototype could be used to motivate workers. The allocation of work also needs to be improved to make sure that workers have the right skills.

[3]`https://products.office.com/en-US/sharepoint`, last accessed on: 2015-08-24

2.3.6. Process-Aware Support for Collaborative Knowledge Workers

Mundbrod et al. [MKR13] introduced characteristics and dimensions of KiPs based on three collaborative knowledge work uses cases. In the first use case the development of an embedded system in the automotive sector is described. Engineers of various different disciplines collaborate to develop complex mechatronic systems. The results are organized based on a best practice approach like the V-model that provides them with an overview about the progress. In the second use case criminal investigations are studied that gather factual information to answer questions or solve problems. Investigators have to determine which standard procedures can be applied, e.g., securing of evidence. The third use case is concerned with complex financial service requests to handle combinations of products from customers. Related use cases are insurance claim handling, intensive patient care and customer onboarding. The authors conclude that all three use cases have in common that they require human assessment and decisions from experts, continuously growing information and handling of unpredictable situations. Figure 2.15 illustrates commonalities in different use cases for collaborative knowledge work.

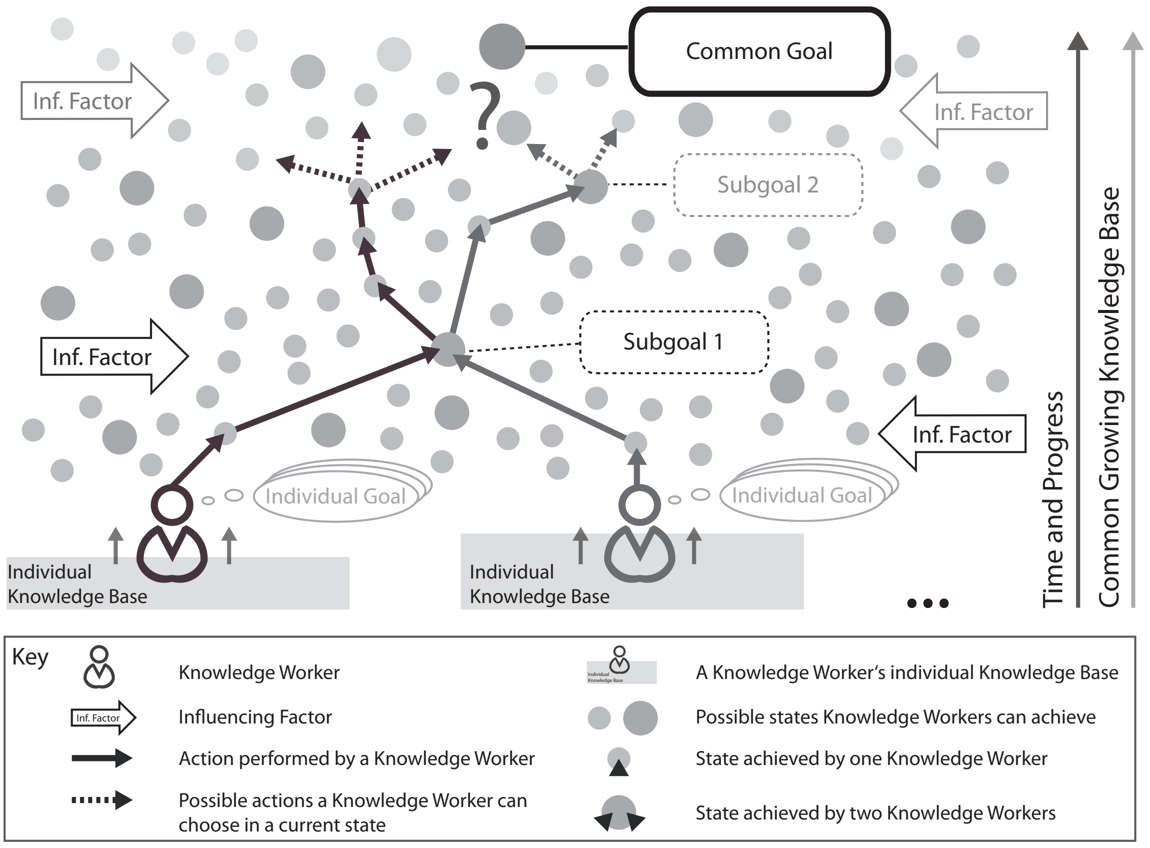

Common Goal
Inf. Factor
Inf. Factor
Inf. Factor
Inf. Factor
?
Subgoal 2
Subgoal 1
Individual Goal
Individual Goal
Individual Knowledge Base
Individual Knowledge Base
...
Time and Progress
Common Growing Knowledge Base
Key
Knowledge Worker
Influencing Factor
Action performed by a Knowledge Worker
Possible actions a Knowledge Worker can choose in a current state
A Knowledge Worker's individual Knowledge Base
Possible states Knowledge Workers can achieve
State achieved by one Knowledge Worker
State achieved by two Knowledge Workers

Figure 2.15.: Commonalities of uses cases for collaborative knowledge work [MKR13]

Based on these use cases the authors present four characteristics (C1 - C4) for collaborative knowledge work. C1 is concerned with *uncertainty* that results from unpredictable influence factors that are intertwined with dynamic correlations. This requires a continuous assessment of planned and finally conducted actions with feedback loops. The course of action needs to be determined dynamically by the involved knowledge workers, i.e., they have to be empowered to structure the KiPs. C2 refers to the *goal orientation* in collaborative knowledge work that is an integrative factor for knowledge workers. Due to characteristic C1, goals are incrementally divided into sub-goals that can be achieved in a shorter period of time. While goals remain stable on the highest level, sub-goals (also known as milestones) can be modified. With C3 the authors mentioned *emergence* as an implication of C1 and C2. Knowledge workers need to continuously adapt to new sub-goals that are incrementally broken down. This agile planning of subsequent next actions to take leads to collaborative knowledge work processes that gradually emerge. Finally, C4 describes the *growing knowledge base* that is based on the interrelation between the progress of KiPs with the advancement of the tacit and explicit knowledge base. Examples for the growing knowledge base are schedules, responsibilities, office documents and e-mails.

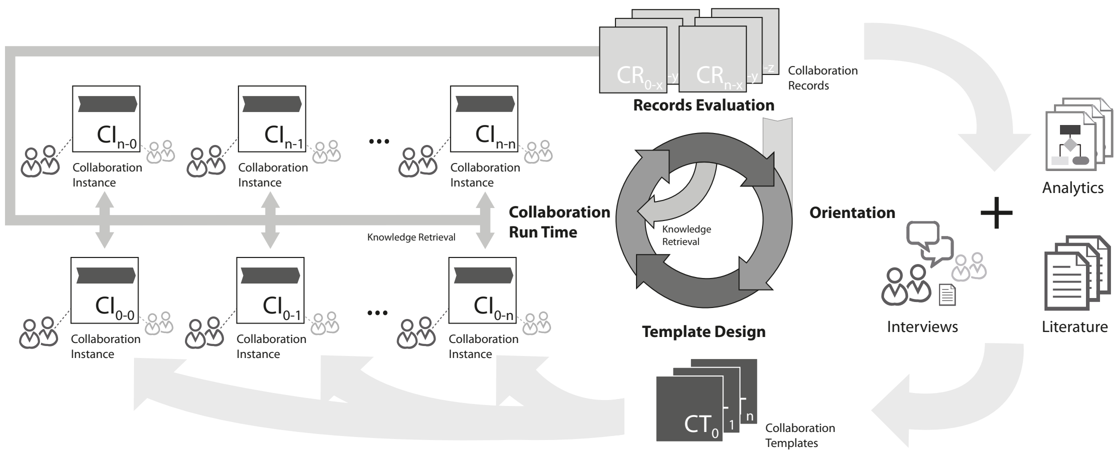


Figure 2.16.: Collaborative knowledge work lifecycle according to Mundbrod et al. [MKR13]

The authors present a collaborative knowledge work lifecycle that needs to be supported by information systems (cf. Figure 2.16). In the orientation phase relevant information necessary to achieve the common goal are gathered through interviews and literature. This information is used to create collaboration templates during the template design phase. During the collaboration at runtime n different knowledge workers work with n different instances of these templates. During this phase they perform adaptations of the initially created template. Finally, the records evaluation phase archives completed instances as collaboration records and evaluates the adaptations of knowledge workers. Potential template improvements are incorporated in the subsequent iteration or already running instances. The proCollab[4] project is still in an early stage and mainly analyzes characteristics of processes for collaborative knowledge work. An initial solution that provides integrated task lifecycle support is presented in [MBR15]. Although proCollab acknowledges the collaborative nature of knowledge work, it provides no incentives that motivate knowledge workers to maintain the collaboration templates and neglects (late) data modeling for KiPs.

[4]`http://www.uni-ulm.de/?id=43398`, last accessed on: 2015-08-24

2.3.7. Case Management Model and Notation

In 2009, the Object Management Group (OMG) issued a request for proposal (RFP) to create a standard modeling notation for case management[5]. Main purpose of this language is to develop a complement of the existing Business Process Model and Notation (BPMN) for data-centric processes that is based on business artifacts [Hu11a, Va11, Bh07]. The final version of the result for this RFP was published in May 2014 as the Case Management Model and Notation (CMMN). While BPMN builds on an imperative process model, CMMN is designed to support declarative process models. Declarative process model are more suitable for dynamic and knowledge-intensive processes that require high flexibility at runtime [Pe08]. Figure 2.17 illustrates an example CMMN model with the visual elements of the notation that specify a fictive write document example case. In this example the creation of a document is modeled with the necessary steps for writing text, creating and integrating figures, generating references and incorporating reviewers feedback.

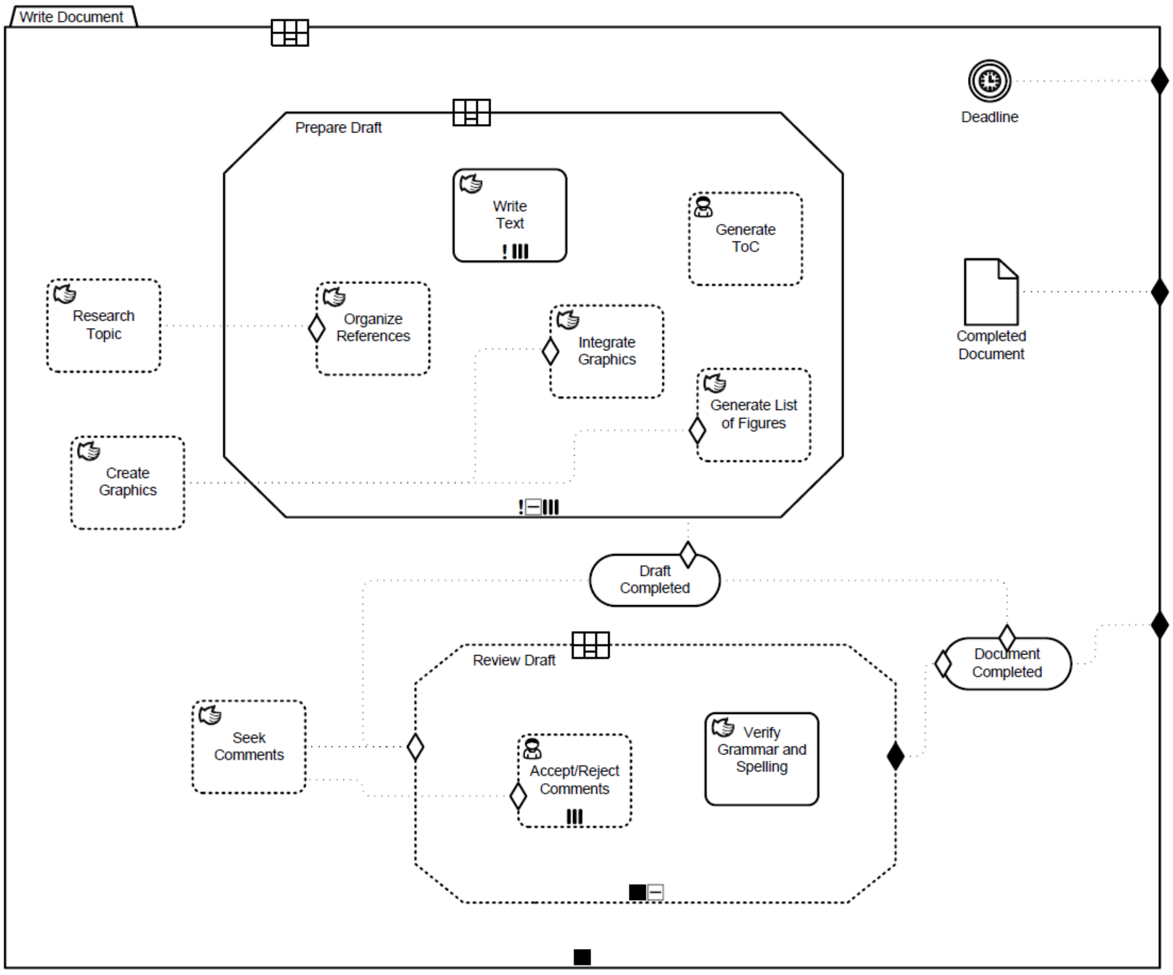
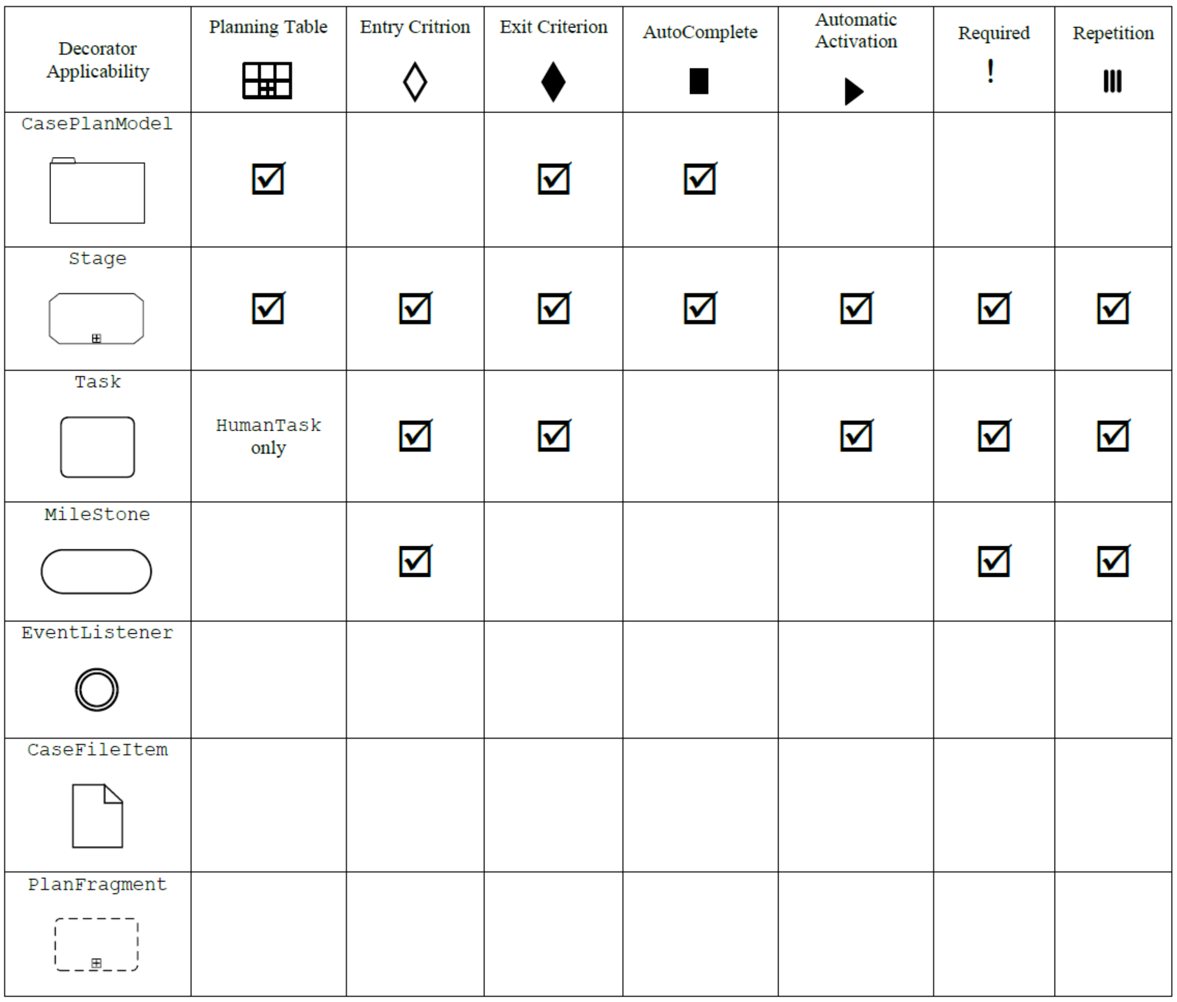


Figure 2.17.: Write document process in the Case Management Model and Notation [Ob14]

[5]http://www.omg.org/cgi-bin/doc?bmi/09-09-23, last accessed on: 2015-05-14

Visual model elements with the decorators that can be applied in CMMN are summarized in Figure 2.18. The *CasePlanModel* is a container for other elements in the case, e.g., write document. *Stages* describe "episodes" that a case can pass through according to the specification [Ob14]. The write document example contains two stages for prepare draft and review draft. *CaseFileItems* represent information objects that directly influence the behavior of the case, e.g., completed document. *Tasks* can be placed in stages and associated to other elements through sentries, e.g., after the research topic task is completed references can be organized. Other elements that can be associated are *EventListeners* and *Milestones*. Main purpose of CMMN is to provide a visual language to model KiPs and a standardized metamodel that allows the exchange of models between different case management tools. This makes CMMN a potential solution regarding the rules and constraints requirements for KiPs described in Section 2.2.1. Nevertheless, a suitable execution environment that fulfills the remaining requirements for KiPs is necessary for CMMN. First and foremost the structuring of KiPs with CMMN would overwhelm no-expert users who are not familiar with this language or process modeling in general.



Decorator Applicability	Planning Table ⊞	Entry Critrion ◇	Exit Criterion ◆	AutoComplete ■	Automatic Activation ▶	Required !	Repetition ‖‖
CasePlanModel	✓		✓	✓			
Stage	✓	✓	✓	✓	✓	✓	✓
Task	HumanTask only	✓	✓		✓	✓	✓
MileStone		✓				✓	✓
EventListener							
CaseFileItem							
PlanFragment							

Figure 2.18.: Applicability of decorators with visual CMMN model elements [Ob14]

We conducted a detailed comparison of requirements of KiPs with CMMN in [MHM15]. Table 2.10 summarizes the results of this comparison based on a fictive example. In the first step the requirements for KiPs are split into two categories. The first category summarizes requirements that are necessary for the modeling environment (M) of a case management solution. The second category is concerned with requirements for a generic execution environment (E). Only 8 of the total amount of 30 requirements for KiPs are assigned to the modeling environment, i.e., the majority of requirements is out of scope for the modeling environment and needs to be fulfilled by an appropriate execution environment. Depending on the specific application scenario the assignment of requirements to the modeling and execution environment might deviate. In the column for the execution environment a diamond symbol is used to indicate whether the CMMN modeling environment can add additional support. These requirements are primarily implemented in the execution environment, but the usage of the CMMN modeling environment can add support. In our research we are primarily interested to engage knowledge workers in the collaborative structuring of KiPs in the execution environment. Although the implementation of the modeling environment uses CMMN to demonstrate the feasibility, the approach presented in this thesis could use other process modeling notations for the modeling environment.

Data modeling is a requirement that needs to be implemented in the modeling environment, whereas CMMN only provides some support through it's flexible information model. It allows the definition of folders with arbitrary data. Late data modeling is related to the execution environment, but CMMN might add support for this requirement through its ability to define basic data structures during execution. The remaining data requirements need to be implemented by the execution environment. Main strength of using CMMN in the modeling environment is the coverage of the knowledge action requirements. Data-driven actions are fully supported and it is also possible to help with the late actions modeling through discretionary items that can be added at execution time. CMMN only provides basic support for rules and constraints through entry and exit criterion's, but it is not specified how they have to be defined. Although CMMN is missing an explicit concept to model goals in the cases, it still provides some support for this requirement because of the milestone concept. Milestones can be considered as sub goals that have to be achieved. Similar to the previous requirement categories, late modeling needs to be provided by the execution environment.

Only one requirement of the process category needs to be primarily implemented by the modeling environment. This requirement is concerned with different modeling styles for different degrees of structuredness. It is not possible to model unstructured knowledge entities with CMMN. As a workaround this requirement needs to be implemented by the execution environment. Only the resource and skill modeling requirement in the knowledge worker category is assigned to the modeling environment. For this requirement CMMN can only provide basic support with the role concept. The modeling of events is supported in CMMN through the ability to define event listeners for the two predefined human and timer events. As a result of this analysis 5 out of 8 requirements for the modeling environment are supported by CMMN. In combination with an execution environment that fulfills the remaining 22 requirements, CMMN seems to be suitable notation to support KiPs. We expect that future versions of this recently published standard will improve some of the current limitations. To the best of our knowledge there are currently no solutions available that provide appropriate execution environments for CMMN.

Requirement	**M/E**	**Modeling Environment**	**Execution Environment**
Data			
R1 Data modeling	M	◐	
R2 Late data modeling	E	◐	⧫
R3 Access to data	E	○	⧫
R4 Access to shared data	E		⧫
R5 Date model comprehensibility	E		
Knowledge Actions			
R6 Data-driven actions	M	●	
R7 Late actions modeling	E	●	⧫
R8 Knowledge action roles	M	◐	
R9 Late knowledge action roles	E	◐	
Rules and Constraints			
R10 Rules and constraints	M	○	
R11 Late constraints	E		⧫
Goals			
R12 Goal modeling	M	◐	
R13 Late goal modeling	E		⧫
Processes			
R14 Different modeling styles	M	–	⧫
R15 Visibility of knowledge	E		⧫
R16 Flexible execution	E	◐	⧫
R17 Unanticipated exceptions	E		⧫
R18 Migration of instances	E		⧫
R19 Learning from event logs	E		⧫
R20 Learning from data	E		⧫
R21 Recommendations	E		⧫
Knowledge Workers			
R22 Resource/skill modeling	M	○	
R23 Workers' interaction	E		⧫
R24 Workers' privileges	E	○	⧫
R25 Late resource modeling	E		⧫
R26 Late privileges modeling	E		⧫
R27 Workers' decisions	E		⧫
R28 Knowledge worker encouragement	E		
Environment			
R29 Model external events	M	◐	
R30 Late modeling of events	E		⧫

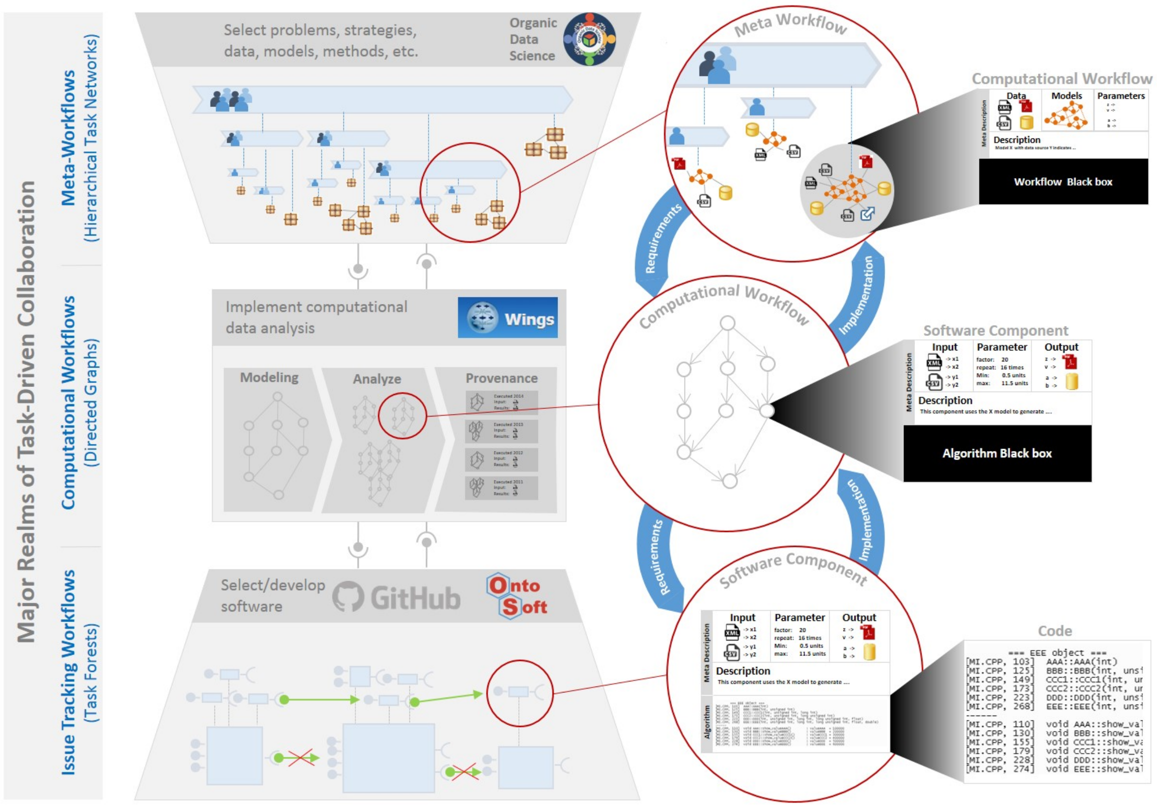
Description for icons:

●	CMMN provides full support	–	CMMN provides no support
◐	CMMN provides some support	⧫	CMMN can add support
○	CMMN provides basic support		

Table 2.10.: Assignment of requirements to modeling and execution environment [MHM15]

2.3.8. Organic Data Science Wiki

Collaboration is a major aspect of science and occurs at many different levels. In [Gi15b], we present a task-centered framework for computationally-grounded science collaborations that support scientists across all levels. Figure 2.19 illustrates the layers of collaboration in this approach. Starting from the bottom of the figure *issue tracking work workflows* support the development of software. Coding sharing sites like GitHub[6], Jira[7] or OntoSoft[8] can support these activities very well. These collaborative activities lead to a task forest that consist of separate issues. On the next higher level *computational workflows*, scientists collaboratively develop computational workflows that implemented a data analysis method. In this approach the scientific workflow system WINGS[9] is used. In [HGL11], we present how this system can be used to support efficient and effective data analysis workflows. On the highest level *meta-workflows* support the activities related to agreeing on joint research questions, developing solutions to these questions and investigating how to get data for the validation of the solution. The activities in these workflows are organized as hierarchical task network, whereas each task is incrementally decomposed into subtasks. These three layers of collaboration are implemented in separate systems but integrated through interfaces with each other.



```
      === EEE object ===
[MI.CPP, 103]  AAA::AAA(int)
[MI.CPP, 125]  BBB::BBB(int, unsi
[MI.CPP, 149]  CCC1::CCC1(int, ur
[MI.CPP, 173]  CCC2::CCC2(int, ur
[MI.CPP, 223]  DDD::DDD(int, unsi
[MI.CPP, 268]  EEE::EEE(int, unsi
------
[MI.CPP, 110]  void AAA::show_val
[MI.CPP, 130]  void BBB::show_val
[MI.CPP, 155]  void CCC1::show_va
[MI.CPP, 179]  void CCC2::show_va
[MI.CPP, 228]  void DDD::show_val
[MI.CPP, 274]  void EEE::show_val
```

Figure 2.19.: Major realms of task-driven collaboration in Organic Data Science [Gi15b]

[6]`http://www.github.com`, last accessed on: 2015-06-23
[7]`http://de.atlassian.com/software/jira`, last accessed on: 2015-06-23
[8]`http://www.ontosoft.org/`, last accessed on: 2015-09-07
[9]`http://www.wings-workflows.org`, last accessed on: 2015-06-23

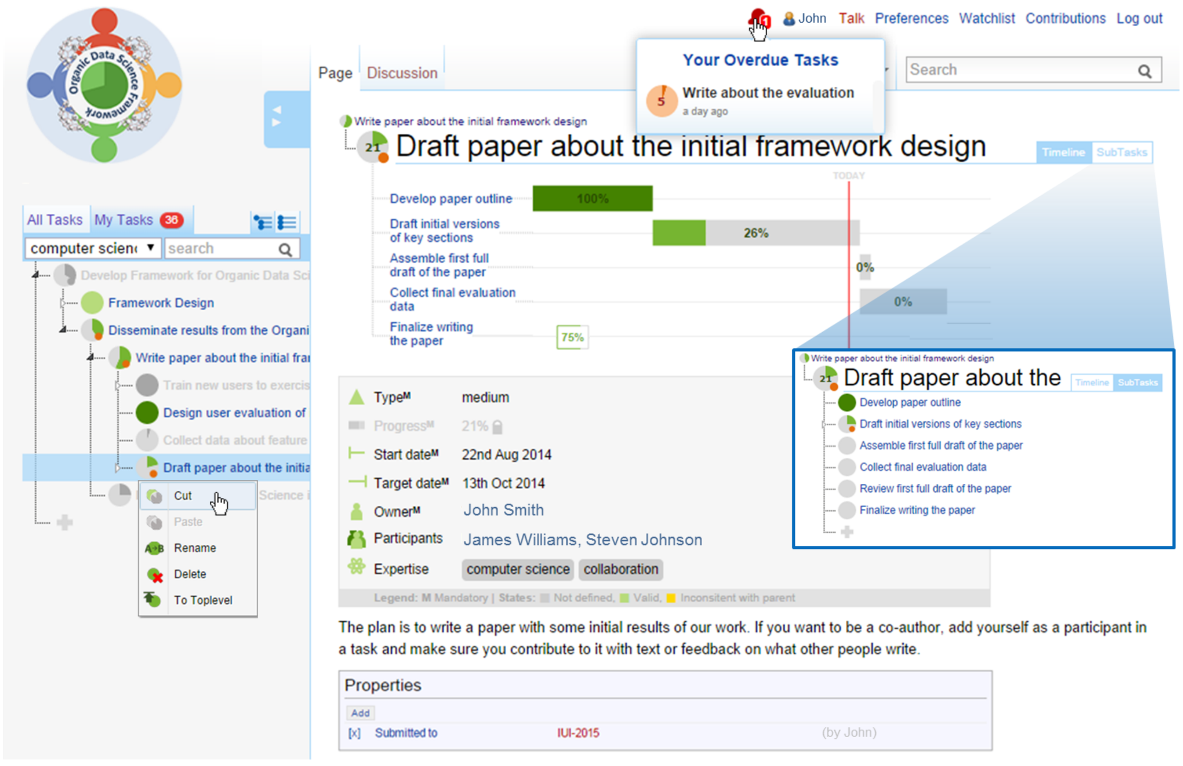
Science has become an increasingly collaborative endeavor with many researches contributing to answer complex scientific questions. Scientific collaborations often revolve around sharing instruments, shared databases and around a shared scientific question. In [Mi15b], we proposed the Organic Data Science[10] approach as virtual crowdsourcing community for open collaboration in science processes. Crowdsourcing is defined as the transformation of tasks that were performed by internal employees of an organization to the crowd through an open call [Ge11]. In science, crowdsourcing parts of the research process could provide valuable knowledge and resources outside an research organization or university to solve complex scientific questions. The Organic Data Science framework achieves this goal through three key features [Mi15b]:

- **Self-organization:** Tasks are used for the coordination of work in a scientific endeavor and users can create joint tasks, decompose them into smaller tasks and track their progress. Tasks are considered as tool for shared social cognition that considers knowledge not only in individual minds, but also in tools and objects that they share. In science, decomposition of tasks is an important aspect since many procedures exhibit goal-oriented hierarchical structures. Scientists can perform all these steps on their own in a self-organized approach.
- **Sustainable online communities:** Science goals often require active communities that are engaged over a long period of time. Studies from social sciences provide useful design principles for the Organic Data Science framework [Kr12a]. We incorporated social design principles that have proven to be successful with respect to the challenges for building successful online communities introduced in Section 2.1.3. These social design principles are used to design user interface features in the solution.
- **Open science processes:** Science processes are made explicit, so that everyone can comprehend the progress of the collaboration. Open tasks with the required expertise that are necessary to complete the tasks can be queried without any restrictions. New scientists can join the process and participate in tasks that match best to their skills. In order to make the participation for new members of the community as easy as possible, the framework provides a separate training that needs to be undertaking before contributing in the Organic Data Science.

Figure 2.20 shows a screenshot of the Organic Data Science Wiki that implements the approach. It is developed as an extension of the Semantic MediaWiki platform, which provides an intuitive user interface that hides any formal notation from the user [Br12]. The platform is extended with tasks that are shown on the left-hand side of the wiki. Little pie charts indicate the progress of every task, whereas this progress is automatically computed based on the lower level subtasks. If the progress cannot be automatically derived, then a metadata attribute has to be entered manually. Other metadata attributes are start date, end date, owner, participants and expertise. The expertise summarizes required skills to complete the task. Scientists can collect expertise in their personal profile by completing many tasks. Based on the start and end date a timeline on top of the wiki page is automatically generated. Tasks can have different states depending on their metadata properties, that are described in detail in [Gi15c]. Tasks that are not finished yet although their end date has expired are shown in

[10]`http://www.organicdatascience.org/`, last accessed on: 2015-05-14

top of the wiki as overdue tasks. Results of tasks are documented on the wiki page as text or attachments in the structured properties of the Semantic MediaWiki.


John
Talk
Preferences
Watchlist
Contributions
Log out
Page
Discussion
Search
Your Overdue Tasks
Write about the evaluation
a day ago
All Tasks
My Tasks
computer scien
search
Develop Framework for Organic Data Sci
Framework Design
Disseminate results from the Organi
Write paper about the initial fra
Train new users to exercis
Design user evaluation of
Collect data about feature
Draft paper about the initia
Cut
Paste
Rename
Delete
To Toplevel
Write paper about the initial framework design
Timeline
SubTasks
TODAY
Develop paper outline
Draft initial versions of key sections
Assemble first full draft of the paper
Collect final evaluation data
Finalize writing the paper
100%
26%
0%
0%
75%
Review first full draft of the paper

Draft paper about the initial framework design

Type	medium
Progress	21%
Start date	22nd Aug 2014
Target date	13th Oct 2014
Owner	John Smith
Participants	James Williams, Steven Johnson
Expertise	computer science collaboration

Legend: M Mandatory | States: Not defined, Valid, Inconsistent with parent

The plan is to write a paper with some initial results of our work. If you want to be a co-author, add yourself as a participant in a task and make sure you contribute to it with text or feedback on what other people write.

Properties

[x] Submitted to	IUI-2015	(by John)

Figure 2.20.: Screenshot of the Organic Data Science representing a task as wiki page [Mi15b]

Science can be classified as KiP since it fulfills all the characteristics introduced in Section 2.2. In particular the user interface design based on social design principles for successful online communities is an interesting approach to motivate knowledge workers. We expect that the motivation of knowledge workers has a positive impact on the structure of KiPs. Nevertheless, the approach has some limitations that we seek to improve in this thesis. In the Organic Data Science Wiki the process always emerges ad-hoc, i.e., it is not possible to define and incrementally improve templates for recurring KiPs. It is not possible to explicitly assign tasks to execution roles so that they are not visible to everybody, which is an important requirement in many organizations. Dependencies between tasks cannot be defined resulting in large task trees, which makes it difficult for knowledge workers to foresee suitable next steps in the process.

Although structured properties of the Semantic MediaWiki can be used to structure information in the Organic Data Science Wiki, it is not possible to automatically update the progress of tasks when mandatory properties are entered. This might lead to increased effort for the maintenance of metadata properties for tasks. Similar to the aforementioned projects, the Organic Data Science is still in an early stage of development and it is continuously improved with feedback of scientists that use they system in pilot projects. In future work there will be an integration of the Organic Data Science Wiki with a scientific workflow system that can be used to conduct and document computational experiments with provenance information [HGL11]. Scientific data in the structured properties of the tasks could be submitted to the scientific workflow system for computation. Figure 2.21 shows the evolution of a community with geoscientists working together on a journal composed of geoscience papers of the future (GPF) that is described more detailed in [Gi15b].

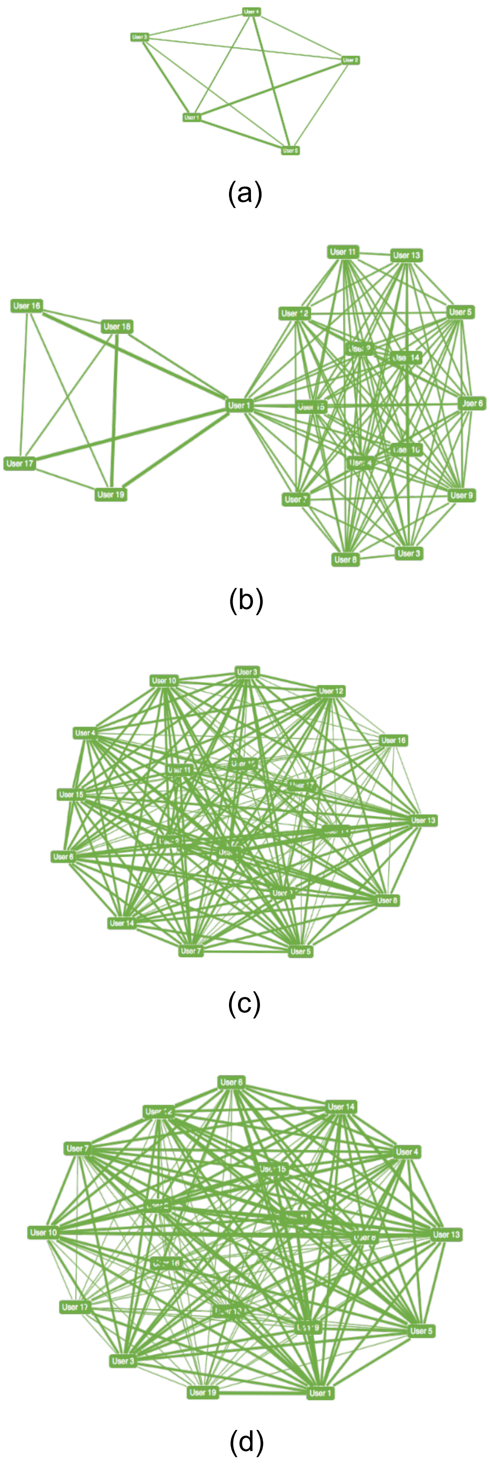


Figure 2.21.: Evolution of the GPF community [Gi15b]

If you can't describe what you are doing as a process, you don't know what you're doing.

W. Edwards Deming

CHAPTER 3

Structuring Knowledge-Intensive Processes

This section describes the approach developed in this thesis for the collaborative structuring of KiPs. Based on this approach, the Chapter 4 presents an implemented software solution that will be used for the evaluation of the main hypothesis. Section 3.1 illustrates the emergent structuring of KiPs with the participating roles as well as the evolution from unstructured work plans to reusable work templates. Section 3.2 presents lightweight structuring concepts that are used as metaphors for knowledge workers not being familiar with process modeling notations. It is important to motivate and encourage knowledge workers to structure KiPs with their expertise to ensure high quality work templates. Section 3.3 introduces features that are based on social design principles for successful online communities to engage knowledge workers in the structuring of KiPs. Based on these features and the lightweight structuring concepts for KiPs, Section 3.4 presents the design of generic user interface components for end-users.

3.1. Emergent Structuring of KiPs

One of the characteristics of KiPs introduced in Section 2.1.1 is concerned with emergence, i.e., the structure of these processes accrues through contributions of many involved knowledge workers that contribute with their expertise. Initially, the processes are either completely unstructured or some basic structure might already be provided from previous executions. During the execution of an already mature KiP, changes on the structure might become necessary in case it is not suitable anymore. In Section 3.1.1, the roles participating in the emergent structuring of KiPs are introduced with their responsibilities. Section 3.1.2 describes how KiPs evolve in our approach by extending previous work Neubert made with Hybrid Wikis in his PhD thesis [Ne12]. While Hybrid Wikis were only concerned with the data layer, our work provides a solution applicable to the larger scope of KiPs. In addition to Hybrid Wikis we consider the problem of motivating knowledge workers to contribute to the emergent structuring for KiPs. Based on the experiences with Hybrid Wikis that we gained from several industry and research projects [MN11, Ha13b], we found that the active involvement of end-users is crucial for the success of this approach.

3.1.1. Participating Roles

Since we extend the Hybrid Wiki solution with additional structuring concepts for KiPs, our approach increases the responsibilities of roles participating in the emergent structuring of processes. Similar to the Hybrid Wiki three different roles for visitors, authors and tailors are distinguished in our approach. The original definition of these roles is inspired by [DIZ06]. It is important to note that users in our approach can have multiple roles assigned for different KiPs, e.g., a user might be author for the KiP a and tailor for another KiP b at the same time. Next to participating roles for the structuring of KiPs, it is possible to define functional groups that specify access rights for content as well as roles for tasks.

Visitors are able to browse structured and unstructured content of wiki pages. For this purpose they can use different techniques to view, navigate, search and explore the content. Main purpose of tasks is to coordinate the modification of content within a KiP. Since visitors are not allowed to modify content, they are also not able to see and interact with tasks. Depending on the application scenario the number of visitors might range from none to an unlimited number in extreme cases. KiPs that are concerned with highly confidential content might have only a very limited number of visitors. On the other end, results of an KiP that have been approved for publication can be visible to an unlimited or large number of visitors, e.g., after completion of a research project the final publications related to the project can be accessible for everyone. Furthermore, visibility of content for visitors might be further restricted through fine-grained access rights for content.

Authors are privileged users that are able to modify, link, comment and discuss content that is represented on wiki pages. In our approach authors require no specific modeling or computer science capabilities. Modifications of the content are coordinated through tasks that can be assigned to authors. All modifications are only related to wiki pages. Similar to visitors, the number of authors might vary to a large extent depending on the application scenario. Some KiPs that are concerned with complex problems might require a large number of authors, e.g., open science processes described in [Gi15a]. For this purpose crowdsourcing of KiPs that were originally performed by internal employees in an organization to the crowd through an open call provides manifold new possibilities. Depending on the crowdsourcing paradigm the preselection of authors might be qualification-based, context-specific, both or none [Ge11]. With none everybody is able to contribute to the structuring of a KiP without any restrictions, i.e., there are no visitors. The remaining options require an authorization step depending on qualifications or context to upgrade visitors to authors.

Tailors require modeling capabilities to maintain the schema elements that are used on wiki pages. Although tailors don't have to apply programming languages for the maintenance of schema elements, they have to understand general data and process modeling techniques. While many authors might be involved in KiPs, only a few tailors are usually responsible to maintain the schema for KiPs that are assigned to them. In many application scenarios tailors might also be authors at the same time, i.e., a subset of the author team is responsible for maintenance of the work template. The main purpose of tailors is to uniform, constrain and clean up work templates. Tailors can be supported through process and data mining algorithms that are used to analyze the contributions of authors.

3.1.2. Evolution of Knowledge-Intensive Processes

Highly structured and repetitive processes that are located at the top of the process management spectrum presented in Section 2.1.1 are specified top-down and only few process designers are usually involved. In contrast, the course of action in KiPs is highly uncertain and the structure gradually emerges according to the available knowledge base. Many knowledge workers have to be involved in the structuring of KiPs, which basically upends the prevalent role model participating in the process of process modeling. While Neubert presented an approach for the evolution of data structures [Ne12], we extend this approach with an evolutionary approach for process structures to adequately fulfill the requirements of KiPs. In line with Hybrid Wikis our goal is to provide a limited set of lightweight structuring concepts for process models that do not overwhelm end-users without computer science background. Figure 3.1 illustrates our extension of Hybrid Wikis considering the emergent structuring of KiPs.

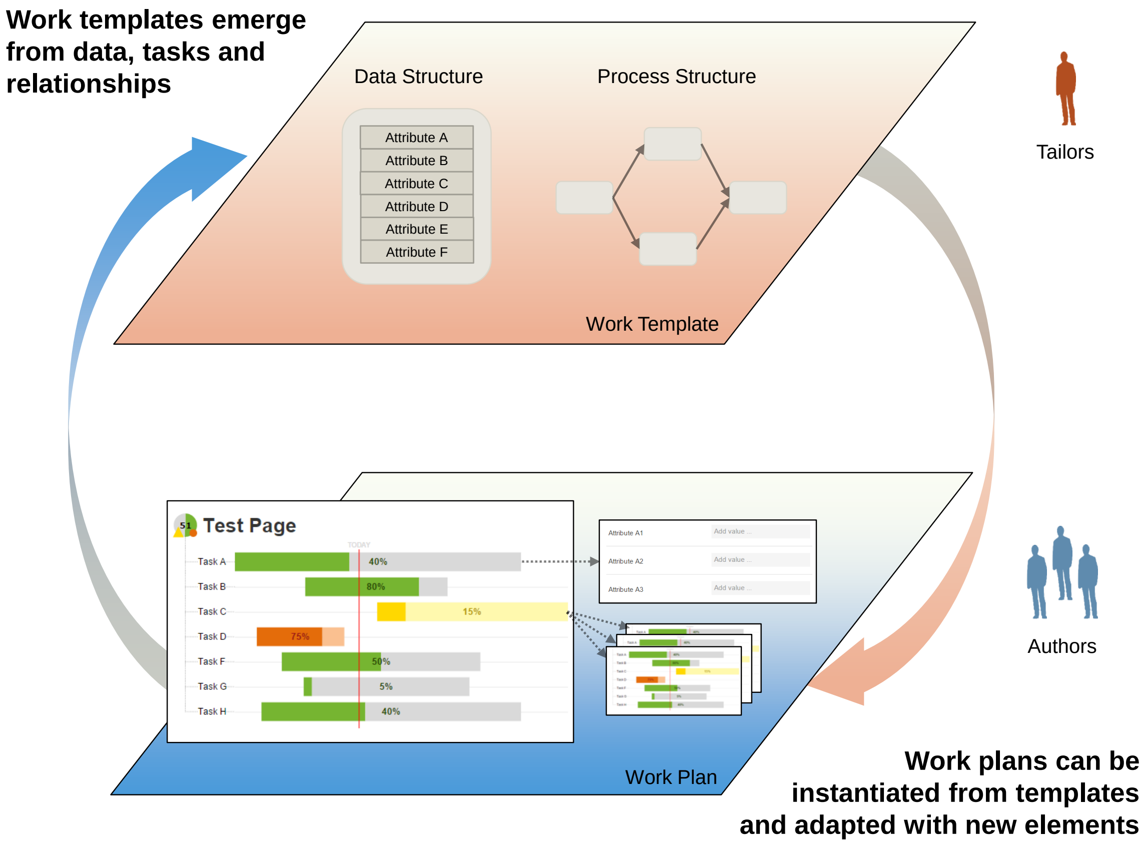


Figure 3.1.: Evolution of KiPs based on an extension of the Hybrid Wiki approach [Ne12]

Authors work with lightweight structuring concepts that define instances of KiPs with data, tasks and relationships. These structuring concepts are visually presented in a way that is easy to understand even without knowledge about process management notations. For this purpose we introduce the notion of *work plans*, which are the central repository to describe the course of action for the fulfillment of goals. Work plans are represented on hierarchically structured wiki pages that can be associated through references. Usually many authors are involved in the structuring of work plans, i.e., similar to the online encyclopedia Wikipedia[1] we envision that work plans are collaboratively defined and maintained through the community. Authors might start using work plans that are completely unstructured, i.e., there are no process and data structures predefined. Unstructured work plans are still valuable for authors as central folder for documents, since they can be used for searching, version control and collaborative editing. Authors can incrementally provide more structure to work plans with lightweight concepts that are introduced in the subsequent section. In case there is already some best practice knowledge available, authors can start working with a predefined *work template*.

Main purpose of work templates is to generalize reusable patterns by making implicit knowledge of (many) experts explicit. These templates are instantiated and gradually improved with every new execution of work plan instances. While work plans provide a lightweight set of structuring concepts that have to be easy to understand for end-users, work templates can be much more sophisticated and take advantage of dedicated process modeling notations, e.g., CMMN. In our approach tailors are responsible for maintaining of work templates and therefore they need to be experienced with process modeling. Nevertheless, tailors need not have programming skills since they should only have to use process modeling languages that abstract from implementation details. Tailors might also be authors at the same time since these roles are not exclusive. Every work template consists of data and process structures that are integrated, i.e., the availability of data elements drives the progress of the process. Tailors analyze adaptations made by authors to identify recurring process and data structure patterns that might be valuable for future iterations of the KiPs. Although we provide a generic user interface for authors that is based on a wiki, we envision tailored interfaces for specific domains in the future. Interfaces for tailors can be much more standardized since they are based on standards like CMMN.

Our evolutionary approach provides various advantages for the support of KiPs. First, knowledge workers are involved in the structuring of KiPs as authors through lightweight concepts. This is an important requirement for future support of KiPs since it is impossible for tailors to predefine their structure entirely. Second, implicit knowledge of experts is made explicit and reused for the organizational knowledge creation. This makes it easier for less experienced knowledge workers in an organization to gather procedural knowledge. One could imagine that proven best practice knowledge that is currently captured in generic frameworks becomes much more actionable. Third, our approach could become an enabler for new process and data mining techniques since existing approaches are only based on implicit patterns in event logs. The power of big data analysis can be exploited much better based on explicit patterns that are structured by end-users. Fourth, knowledge workers are not restricted by rigid and strict processes which limit their freedom to generate new innovative solutions for complex problems. Advantages of agile methodologies for unpredictable situations are combined with the benefits of process management, e.g., traceability and reproducibility [We12].

[1]`https://www.wikipedia.org/`, last accessed on: 2015-09-02

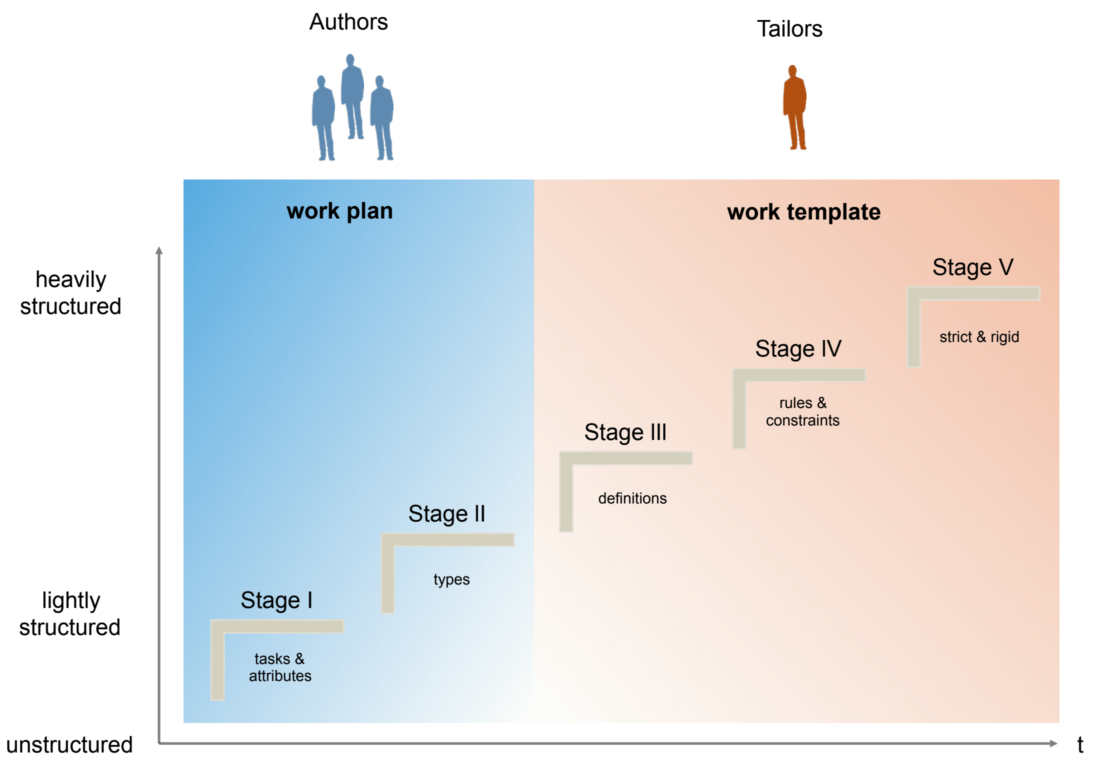


Figure 3.2.: Extended degree of structure from the Hybrid Wiki [Ne12]

Based on our evolutionary approach the degree of structure for work plans and templates is illustrated in Figure 3.2. Although our goal is to enable an emerging structure, it is not necessary for KiPs to pass through every stage, i.e., some work plans might never be generalized with templates or they might remain on one stage. Too much structure could restrict knowledge workers in their freedom and therefore the highest stage is not always the optimal solution. The lowest level only consists of unstructured content and is called stage 0 (not shown in Figure 3.2). *Stage I* describes the first level of structure that extends content with attributes and tasks that are added by authors. In this stage attributes can already be assigned to tasks as mandatory deliverables. The next level of evolution is *Stage II* with an additional type that is attached to work plans. The first two stages are structured by authors in an own user interface, while the subsequent stages are maintained by tailors. *Stage III* introduces definitions for the lightweight structuring of elements. In the simplest case, work templates only consist of definitions that group tasks and attributes for a type together. *Stage IV* introduces constraints on attributes to ensure proper attribute values, e.g., visual warnings in the user interface for authors to advert to multiplicity constraints, missing or wrong values. Constraints on tasks are introduced to specify logical relationships between tasks, i.e., some tasks are only enabled after completing certain other tasks. In this stage authors are still able to skip tasks or delete attributes in case they are not necessary. *Stage V* introduces strict and rigid attributes, i.e., these attributes and associated tasks cannot be deleted in the work plans to enforce mandatory steps in KiPs. Depending on the application scenario this might not always be desirable, so that the structuring might be finished on a lower level.

3.2. Lightweight Structuring Concepts

In this section, we introduce the main concepts for collaborative structuring of KiPs. For this purpose we extend the data structure concepts of Hybrid Wikis presented in [Ne12] with lightweight concepts for the process structure. Figure 3.3 illustrates these structuring concepts with color codes to distinguish data structuring concepts of the Hybrid Wiki from process structure concepts introduced in this thesis. Concepts in gray color are used to describe the data structure layer that is extended with orange concepts for the process structure layer. Wiki pages are located in wikis and they contain unstructured content that is described in a markup language. Every wiki page has a unique Uniform Resource Locator (URL) and name that is unique within the wiki. Attributes can be assigned to wiki pages to add structured content. Wiki pages that describe semantically the same content can be tagged with a common type. This flexible information structuring mechanism is extended to adequately support KiPs.

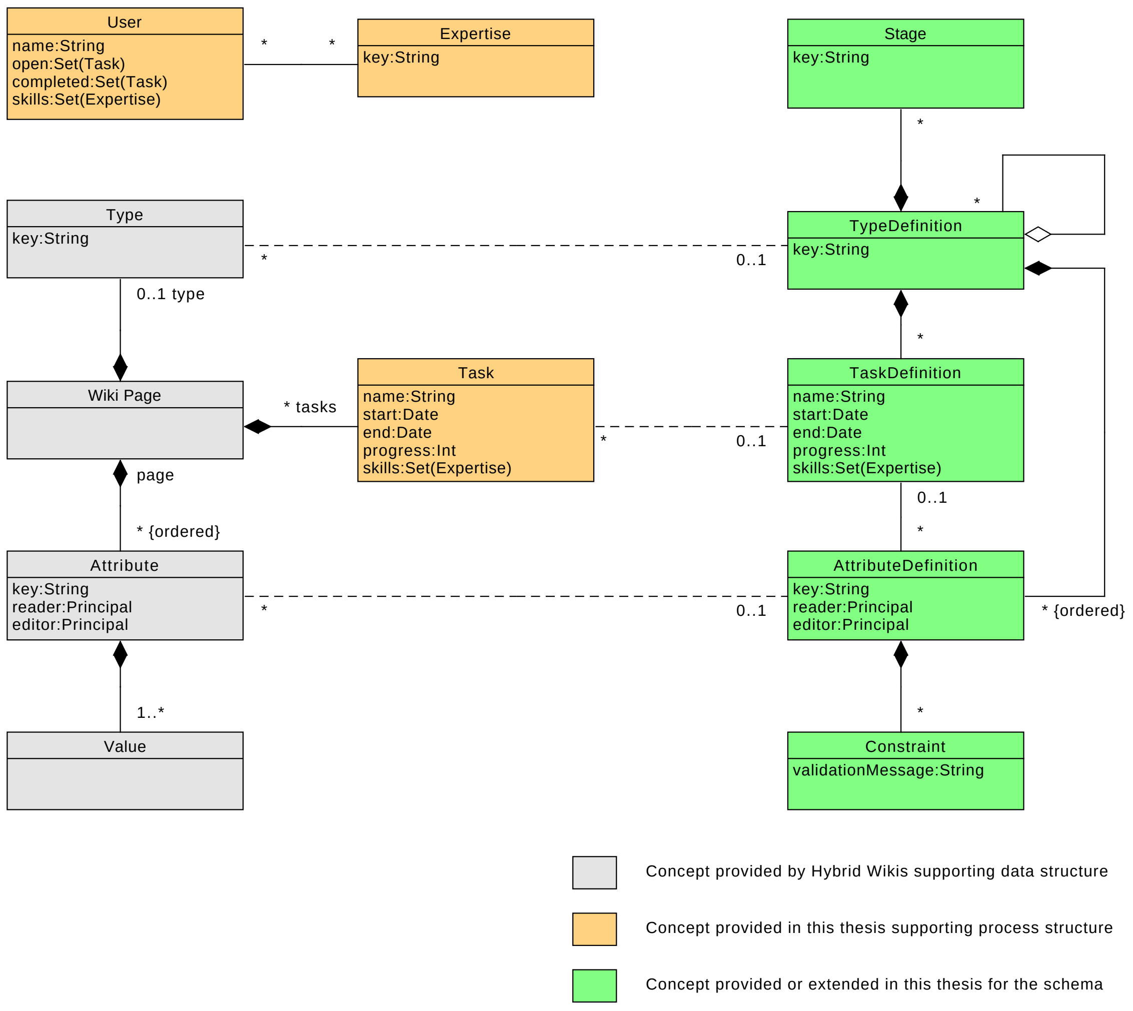


Figure 3.3.: Overview of the main concepts for structuring KiPs within a wiki

Orange and gray structuring concepts are visible in the user interface for authors to describe work plans. Section 3.3 introduces the user interface features for these concepts that are developed in this thesis. Work templates are based on green structuring concepts that are used by tailors to describe the schema. Due to the integration of the data and process structure several extensions are necessary on the schema level. Structuring concepts for work plans and templates are maintained independent from each other to facilitate the emergent structuring of KiPs presented in Section 3.1.

3.2.1. Expertises

Tasks in KiPs are used to coordinate work to suitable users in an organization, i.e., users that have the right competencies to complete the tasks. Authors can create an *expertise* as structured concept to describe a skill that is necessary at least once in a KiP. Expertises can be considered as a simple text snippet or tag attached to tasks. Figure 3.4 illustrates the relationships of expertises in our approach more detailed. Every task may have an arbitrary number of expertises assigned, e.g., expertises a, b and c are necessary to complete this task. Users that complete tasks automatically receive the expertises that were assigned to this task. Thereby, users can collect expertises to document their skills in an explicit way. Main advantage of this concept is that the work allocation between open tasks and available resources can be improved through the explicit structuring of expertises. Users within a KiP can search for tasks that match with their skills or suitable candidates for a task could be automatically retrieved.

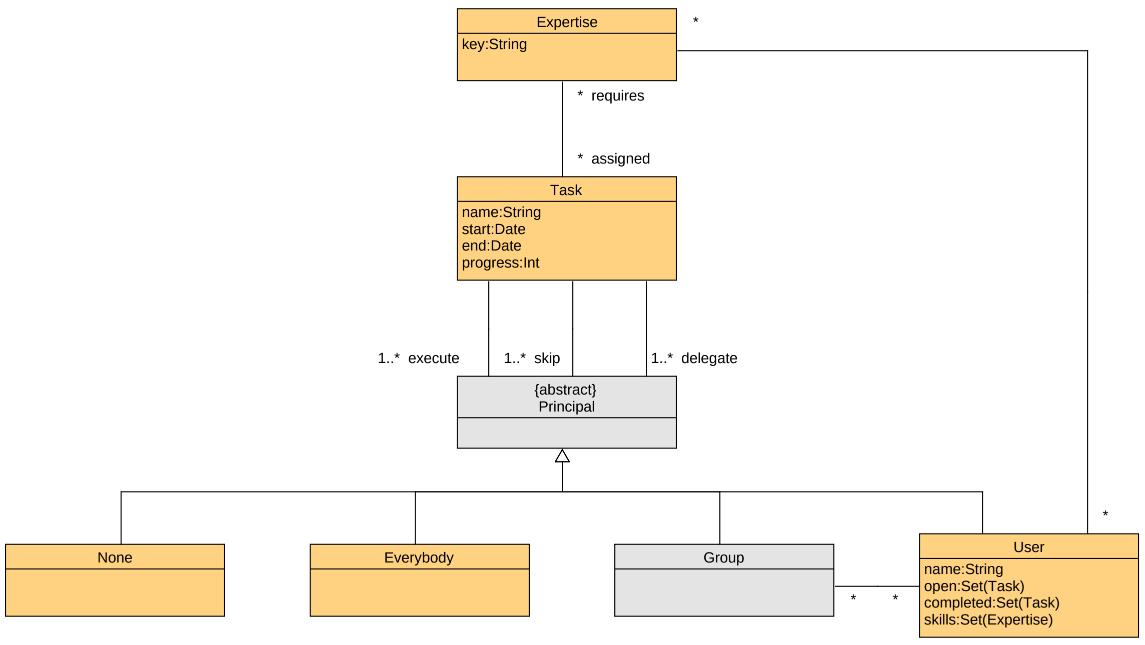


Figure 3.4.: Relationships of expertises and tasks with the role model

3.2.2. Users

Users are central concepts in KiPs since they are responsible for fulfilling goals. These goals are incrementally broken down by users from a very abstract level to manageable junks of work. The latter, in turn, are represented as tasks as soon as they can be accomplished in one coherent step by a certain group of users. Additional concepts are necessary to support tasks in KiPs. Figure 3.4 introduces these concepts with *principals* as an abstract concept. Every task has three roles assigned to at least one principal. The *execute* role can be compared with the authorization role in workflow management solutions. Principals with this role assigned are responsible for the completion of the task. Next to this authorization role our approach supports additional roles for the distribution of tasks. The *skip* role is used for tasks that can be omitted by the assigned participants, i.e., the task is immediately completed even if not all required actions are finished. Finally, the *delegate* role is assigned to principals that may forward the task to other participants or redo an already completed task. These roles are not distinct and principals can be assigned to more than one role in the context of the same task, i.e., a participant might have the roles to execute and delegate the task.

In our approach four different principals are distinguished that can be assigned to the three task roles. A principal can be an individual *user* in the system that has a set of open and completed tasks assigned. Based on the set of completed tasks it can be automatically determined which skills the user has. Users might also be assigned to an arbitrary number of *groups*, i.e., tasks with their roles can also be assigned to entire groups. Furthermore, we introduced two additional principals for tasks. *Everybody* is used as a technical principal for roles that apply to any user in the system, e.g., a task *a* with execution role *everybody* can be completed by every author without any restrictions. The opposite to the everybody principal is *none* which can also be assigned to all task roles. The none principal is helpful for the roles that are related to the distribution of tasks. In case it is not desired that users are able to skip a task, the nobody role can be assigned as placeholder. The same mechanism can be used to avoid the delegation of tasks.

In addition to the assignment of various roles for tasks, it is possible to allocate principals as reader and editor of attributes, i.e., for every attribute it is possible to specify who is allowed to edit and read it. This is an important requirement for KiPs that can also be found in example scenario A for innovation management processes introduced in Section 2.2.2. In this example, reviewers in an organization create evaluations for ideas that are submitted by employees. These evaluations are represented with attribute values that have appropriate access rights. The result of the evaluations are restricted to commissioners and employees should not be able to read them. This fine-grained authorization model might lead to problems if an author creates a new attribute that is already existing on the wiki page but not visible to this author. In this case the attribute is created redundantly with a warning that is visible to the tailor. The tailor is responsible to resolve this warning and check whether the existing authorization model is not appropriate. Redundant attributes might indicate that an author requires reader or editor permissions for this attribute. In case the authorization model is correct and some attributes are reasonably redundant, the tailors should have an opportunity to ignore the warning.

3.2.3. Tasks

Tasks in our approach are represented with a name and associated to *expertises* that briefly describe necessary skills required to complete the task. In addition, every task has a *start* and *end date* that can be specified by the assigned principals. These dates are useful to estimate the duration of the task. Further, they can be used to generate project plans. The *progress* of tasks can be either automatically computed or entered manually depending on the associations of the task. Every task has to be attached to a wiki page whereby wiki pages might have an arbitrary number of tasks assigned. Tasks may also have an arbitrary number of associated attributes, i.e., not every task needs to have an attribute. The assignment of attributes a to task t means that these attributes a have to be created within the specified start and end dates by the principals in the execution role of t.

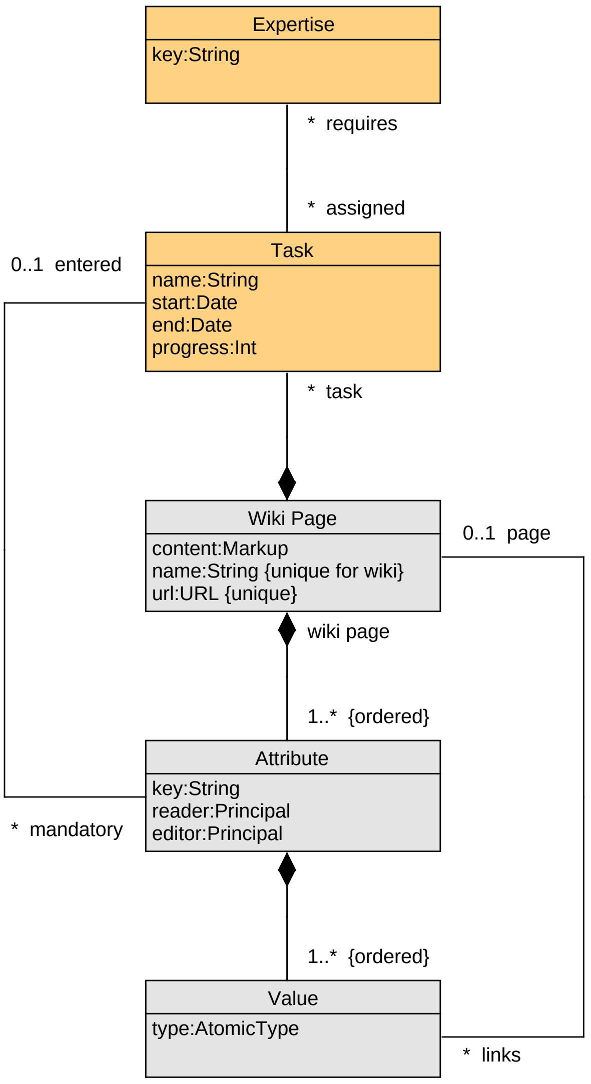


Figure 3.5.: Integration of tasks with the information model

An important requirement for software support of KiPs is the representation of data-driven actions, which we describe in Section 2.2.1. We represent these data-driven actions with tasks that are integrated with the information model in our solution. Figure 3.5 illustrates the relationship between tasks and attributes, whereas there are two options for the completion of tasks conceivable. First, tasks that have no attributes assigned can only be completed by setting the progress manually to 100% or by skipping the task in case the skip role is not specified with none. Tasks that have no explicit attributes assigned might be related to activities that are conducted outside the system. Another possibility could be that no structured data is produced in the involved activities of the task or it is not predictable what kind of data structure is necessary. For this purpose the results of the tasks can be documented in the unstructured content of the wiki page or outside the system. Second, all *mandatory* attributes assigned to the task fulfill the specified attribute constraints. Completed tasks are added to the set of completed tasks of the user and associated expertises are added.

Every wiki page might contain an arbitrary number of tasks to coordinate the creation of structured and unstructured data through users. Based on these tasks it is possible to automatically compute the progress of a wiki page. This progress is used to express how much of the mandatory data on the wiki page has already been created. In combination with the start and end dates for tasks, users can easily comprehend where contributions are missing. We compute the progress of a page p as an average of the progress of all n tasks that are attached to this page with Equation 3.1. We decided to treat every progress equally in the computation of the overall page progress. Main purpose of the progress is to provide a rough estimation for users rather than a computation that is as accurate as possible. Nevertheless, the expressiveness of our metamodel allows to adapt the computation to specific application scenarios. One could use the assigned expertises to weight the impact of tasks on the page progress, i.e., the progress of tasks that require certain expertises have a stronger influence. Another possibility would be to incorporate the start and end dates of tasks, i.e., tasks with a higher duration have a stronger influence on the page progress.

$$page\_progress(p) = \frac{\sum_{i=1}^{n} task\_progress(p.task_i)}{n} \tag{3.1}$$

The progress of the page is computed based on the progress of the attached tasks to this page. In the simplest case the constraints for mandatory attributes are fulfilled as soon as all empty values are entered by the users. In this case the task is directly related to the creation of structured data in the system and completed as soon as all empty attribute values are entered. Attribute values have an *AtomicType* which is described more detailed in [Ne12]. Types of an attribute value can be simple or complex with a link to another internal or external object. Examples for simple attribute value types are date, string, enumeration, integer and boolean. Complex attribute value types link to another structured type that is represented on a wiki page or to an external object, e.g., wiki page p links to another wiki page r of type t. Hierarchical task structures can be represented by assigning attribute values with a complex type to tasks. These complex attribute values are linked to wiki pages that might have tasks again. Thereby, authors are able to create hierarchically organized tasks with wiki pages to incrementally break down very generic goals to executable tasks.

The progress of tasks is automatically computed based on the constraints for the assigned mandatory attributes. Similar to the page progress every mandatory attribute is treated equally important for the computation of the task progress. Depending on the application scenario our metamodel allows to change the impact of certain attributes on the task progress, i.e., certain attribute types like files might have a stronger influence on the task progress than other attribute types like string or date. Tasks that have complex attribute values assigned incorporate the progress of the linked wiki pages into the task progress. The computation of the progress for a task t is given in Equation 3.2 with the number of mandatory attributes $attr_m$ and the number of completed attribute constraints $attr_c$. The page progress for all n wiki pages that are linked to the task are computed recursively, i.e., the equation is repeated for every task in the hierarchy of t. The resulting structure of the task hierarchy is a tree with leaf tasks that have either no attributes or only attributes with a simple type assigned. Due to this structure the *task_progress* equation always terminates at the leafs of the tree.

$$task\_progress(t) = \frac{attr_c * 100 + \sum_{i=1}^{n} page\_progress(t.page_i)}{attr_m} \tag{3.2}$$

The computation of the task progress with Equation 3.2 is always applied for the page process except for tasks that are skipped. Users can skip tasks in case they or one of their groups in which they are member are authorized to perform this role. Skipped tasks are marked as completed although their progress is not automatically set to 100%, i.e., they are removed from the user's worklist. Usually skipped tasks will have no progress since they are immediately skipped in a work plan. Nevertheless, they can be started and skipped at a later stage due to the unpredictable characteristic of KiPs if this is desired within a work plan. Skipped tasks are removed from the computation of the overall page process independent of their current progress value. In case there are pages with tasks assigned this step is repeated for all tasks below the skipped task in the tree. Without removing skipped tasks the page process could never be completed and this might be confusing for users.

The delegate role has no influence on the computation of the page progress, i.e., delegated tasks retain their progress for the new users that are responsible for the delegated task. Delegated tasks can be associated to task structures on other wiki pages and changing their progress might not be desired. In addition, it could be useful for users in the new execution role to have insights about preliminary work related to the delegated task. Finally, the redo role can also impact the computation of the task progress in case an already completed task needs to be enabled again. In contrast to delegate and skip operations that have no influence on the page progress, redo changes the task as well as the page progress. The progress of a task r that is redone will be set to 0% and r appears in the executing user's worklist again. In case task r has associated pages, all tasks linked by r are recursively redone as well and their progress is set to 0%. The progress of tasks above r needs to be updated based on Equations 3.2 and 3.1. We distinguish between a real-time mode that executes the computation with every change of one progress and a batch model that initiates the computation in fixed time intervals. In order to avoid many computations within large hierarchical task structures our metamodel stores the current progress of tasks.

3.2.4. Task Definitions

End-users are empowered to structure work plans with tasks, attributes and types that are created and linked with each other at runtime. The creation of these structuring concepts is not restricted in the early stages and they can be added at any time assuming that end-users have author rights. Work plans can be based on templates that already contain predefined and more sophisticated structuring concepts based on best practices. Usually these templates emerge from the structuring concepts provided by end-users. Work templates consist of definitions that are loosely coupled with the work plan to enable this emergent structuring. Figure 3.6 illustrates the main concepts for the description of the schema that is captured in work templates. *Wiki pages* are structured with *attributes* and *tasks* in addition to unstructured content that is described on the page. Attributes can be assigned to tasks as *mandatory* structured content that has to be created. Every attribute can have one task assigned at most, whereas the number of attributes is not limited for tasks.

The *type* of a wiki page is used to condense pages that are concerned with semantically similar content. Types are represented with a text string that has to be unique within a wiki. Based on this type tasks and attributes of the wiki page can be dynamically determined, e.g., wiki pages of type *project* contain an attribute *description* as string as well as a task for this attribute named *describe goals of the project*. The attribute description is assigned to the task as mandatory attribute, i.e., to complete this task the attribute value has to be entered. Depending on the degree of structure this type is provided by authors (in Stage I and II) or a work template for *projects* can be instantiated (above Stage III) which already contains a predefined structure.

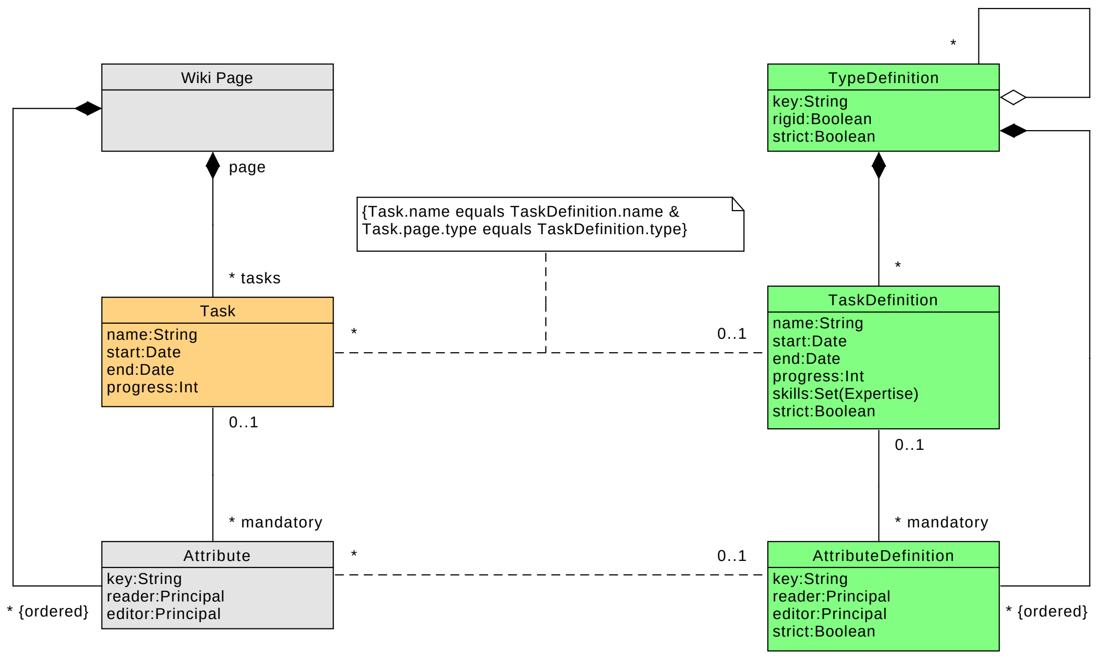


Figure 3.6.: Main concepts for the description of the schema in work templates

In case a work template for *projects* is already maintained by tailors, attributes and tasks are automatically created once the type is assigned to wiki pages. For this purpose the type of the wiki page is compared with existing *TypeDefinitions* based on the provided key in the simplest case. More advanced approaches could be applied to compare structured and unstructured content provided by authors with predefined type definitions, e.g., based on cosine similarity [Hu08]. For example, wiki pages that often contain the term project in the text are likely to be related to the TypeDefinition *project* even though no type is provided by the authors. TypeDefinitions can be structured hierarchically and they contain additional attributes to manage the evolution of the instances. *TaskDefinitions* are assigned to TypeDefinitions, i.e., all wiki pages with the corresponding type also receive tasks with the same names like the TaskDefinitions. The same accounts for attributes that are instantiated on wiki pages based on *AttributeDefinitions* with their association to the same TypeDefinition. Finally, the association of *mandatory* AttributeDefinitions for TaskDefinitions is maintained in the work plan, in which mandatory attributes are assigned to tasks. Main advantage of this approach is that work plans and templates can be maintained independent of each other, which gives authors much more flexibility to perform adaptations on work plans that are important for many KiPs.

Tasks are automatically completed based on Hybrid Wiki constraints that are defined for the assigned attribute definitions (cf. [Ne12] for an overview of possible constraints). Attributes may have several constraints assigned, i.e., it is possible to specify that an attribute needs to fulfill more than one constraint. Hybrid Wikis distinguish between two different kinds of constraints. *Multiplicity constraints* are used to specify how many values an attribute needs to have, e.g., at-least-one, at-most-one and exactly-one. *Data type constraints* validate whether the data type of attribute values is equal to a given type. It is possible to apply data constraints for simple data types, e.g., string, integer, date, boolean, as well as enumerations and references to other wiki pages. For example a data constraint of type string validates that only attribute values of type string are entered. Once all constraints of the attributes are fulfilled, the assigned tasks for these attributes are automatically completed. In our approach we apply data type constraints as default for attributes that are assigned to tasks in the work plan, i.e., for the completion of tasks it is only necessary to provide a value of a given type for the attributes. This type is determined during the creation of a new attribute in the work plan or template. More sophisticated constraints could be determined by tailors, e.g., based on values of already terminated work plans from the past.

AttributeDefinitions and TaskDefinitions have an additional attribute that is used to specify whether they are *strict*. This means that work plans can only be saved when they contain no invalid values, i.e., values that are not compliant with the specified constraints for attributes. With the strict option users are always forced to provide valid structured content for attributes and tasks. When applied to a task definition the work plan can only be saved when mandatory attributes that are assigned to the task are not violating constraints. For example using a string for an integer attribute value would violate the string data type constraint. TypeDefinitions can be marked as *rigid* as shown in Figure 3.6. Rigid types consist of strict constraints only and it is not allowed to modify tasks and attributes, i.e., it is not possible to add or remove tasks and attributes within a rigid type. Tasks of rigid types do not have to be completed and empty attribute values are allowed.

3.2.5. Stages

Main purpose of *stages* is to describe the lifecycle of types that are attached to wiki pages. Stages describe episodes wiki pages pass through depending on the completion of tasks [Ob14]. Figure 3.7 illustrates the concepts for the process structure of work templates. Based on the type of the wiki page the process model is determined using the TypeDefinition. The process model is defined for work templates having a degree of structure on Stage III or higher since they require TaskDefinitions. Although stages can be defined without any TaskDefinitions in the work template, their definition provides no additional value on a lower degree of structure. TaskDefinitions can be grouped in stages, i.e., all tasks that are related with the goal of one stage are summarized. For example, a stage *initialize project* could contain tasks that are related to the initial setup of the project. Stages are helpful to organize the process model since they provide the context for tasks. In combination with rules described in Section 3.2.6, all tasks that are grouped in one stage can be enabled at once.

Stages can only be defined by tailors in the work template to organize the process model. With stages, rules and constraints tailors have a very powerful set of modeling concepts to specify data-driven process models for KiPs. It is not possible for authors to define them since much more sophisticated knowledge about process modeling is required to define stages with rules and we don't want to overwhelm non-expert users. Nevertheless, tailors can take advantage of the lightweight structuring concepts provided by authors for the definition of process models. Once a set of tasks is created by authors that are generalized to TaskDefinitions, it becomes much easier for tailors to define stages and rules. Another advantage of this approach is that authors can make adaptations in the work plan. The lightweight structuring concepts are means for authors to handle unpredictable situations which are one of the most challenging aspects in KiPs.

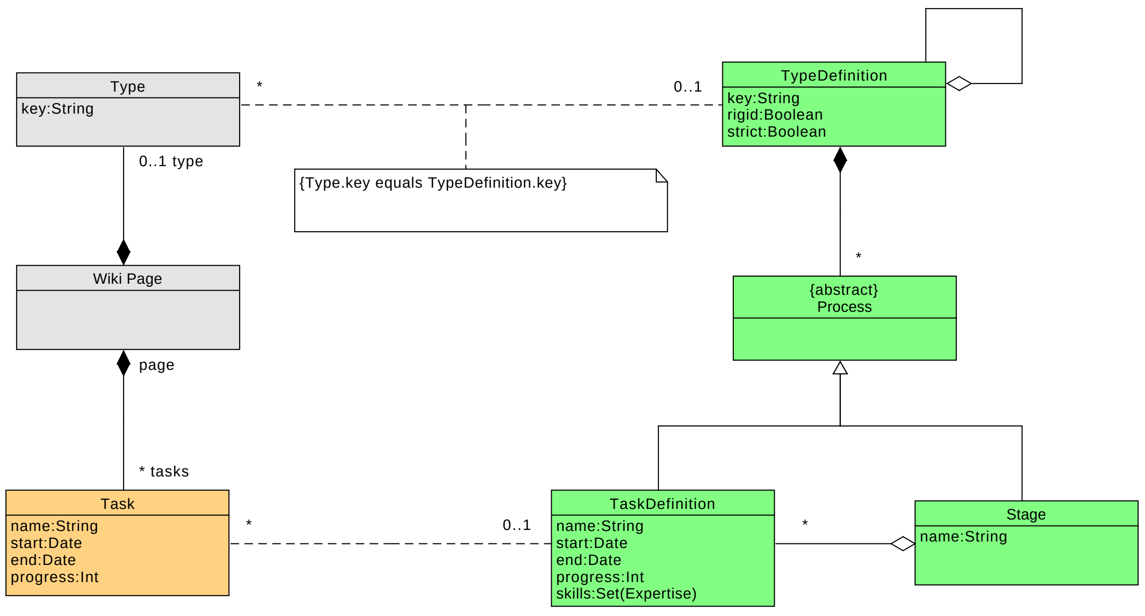


Figure 3.7.: Process model concepts in the work template with their link to wiki pages

3.2.6. Rules

Tasks and stages can be associated with entry and exit *rules* that are maintained by tailors in the work template. The creation of rules requires Stage IV since they are linked to stages and TaskDefinitions. In Section 2.2.1 rules and constraints are introduced as important requirements for KiPs. In our approach constraints are defined on AttributeDefinitions to determine when tasks are completed since data is the main driver for the progress of tasks. In contrast to constraints that link the progress of tasks with the data structure, rules are only concerned with tasks and stages within the process structure. Main purpose of rules is the definition of logical dependencies between atomic tasks and stages that are used to group other tasks. Rules are important for work plans that have many tasks because users could be overwhelmed with the selection of the next task in their worklist. Figure 3.8 illustrates simplified view on the associations of rules with process elements.

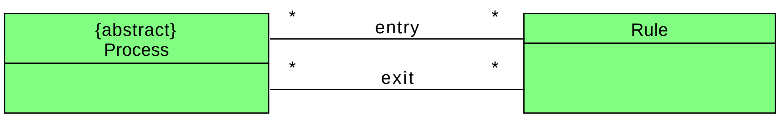


Figure 3.8.: Simplified association between rules and process elements

The producer and consumer pattern is a typical example for a dependency in KiPs that can be represented with rules. In this pattern a task (producer) needs to create some output that is processed by another task (consumer). Our process model allows the creation of basic producer and consumer patterns for tasks and stages. For example a task *t1* needs to be completed before another task *t2* can be started. With rules these consumer tasks are hidden and users are not overwhelmed with tasks that cannot be executed due to logical dependencies. Variations of this pattern are rules with multiple consumers or producers, e.g., more than one task needs to be completed before the subsequent consumer is enabled. In case more than one producer process element is associated two different rules can be applied. Consuming process elements are enabled either as soon as one or all producers are completed. In addition to tasks, rules can also be associated with stages as producer and consumer.

The introduction of rules on process elements can be very valuable to capture frequent work patterns that help to guide users during KiPs. On the downside, too many rules are critical because they can restrict knowledge workers in their decisions and limit their flexibility. Limiting the flexibility is not always desired due to the unpredictable characteristic of knowledge work. In our approach tailors need to find the right balance between flexibility and guidance. It is important to note that our approach just provides the right means to facilitate this balance, but it is equally important that tailors maintain the right degree of structure in the work templates. In some application scenario for KiPs there might be external restrictions that have to be adhered to, e.g., compliance regulations of legal authorities. For example certain issues need to be appropriately documented before the work plan can continue with the other tasks. In this case a high degree of flexibility within certain segments of the work plan for authors is not desired since they might circumvent these restrictions.

3.3. Features Based on Social Design Principles and Patterns

There many examples for communities that became successful through content that is generated primarily by users. Most notable the online encyclopedia Wikipedia[2] consists of a plethora of articles that were edited by volunteer users. More than 35,000 of these users made five or more edits during the month of February 2011 [Kr12a]. Another less prominent example is NASA's Clickworker community[3] that helps scientists to analyze data by clicking Mars photographs to trace the outline of craters. Communities can also become extremely successful in developing software like the famous Apache OSS project[4]. These examples demonstrate how online communities can be used for collective problem solving and the creation of various different types of content. Such communities are powerful because they break the barriers of time, space and scale that are limiting offline collaborations through information technology [Kr12a]. We envision that online communities revolve around work templates in our approach. Although online communities can become very successful, there are also many examples of communities that failed because they were not able to attract and sustain enough members. Therefore, we incorporate findings from social sciences in our solution.

Work templates emerge bottom-up through the lightweight structuring concepts that are introduced in Section 3.2. These structuring concepts are collaboratively maintained by authors on wiki pages and generalized to reusable templates by tailors. Work templates can only be generalized in case enough structure is provided on wiki pages by authors. Therefore, it is crucial that the system attracts as many authors as possible and these authors are also motivated to maintain the structuring concepts on wiki pages. For this purpose we apply social design principles and patterns for successful online communities from social sciences in the design of features for the user interface. The social design principles are based on a subset of the principles that are presented by Kraut and Resnick in [Kr12a]. The selected subset is concerned with principles for the early stages of communities since more advanced principles are less relevant in the current phase of the project. The patterns are based observations of Polymath [Ni12] and lessons learned of ENCODE [Bi12]. Although these patterns are retrieved from platforms that are related to research questions, we are convinced that they are equally practicable for other domains.

The application of these principles and patterns for processes is evaluated with the Organic Data Science framework that we presented in [Mi15b]. In the following, features for the lightweight structuring concepts that are based on these social design principles and patterns are introduced. While the general structuring concepts are developed based on the requirements of KiPs, some additional attributes in our model are necessary to incorporate the design principles and patterns. In particular, expertises for tasks and user profiles are introduced to represent the features related to the specific social design principles and patterns. The structure of the principles is organized with the categories presented by Kraut and Resnick in [Kr12a]. Since we only incorporate principles concerned with early phases of communities, future work could propose additional features for retention of members and for regulating behavior. In the following the six categories are presented with a brief explanation (numbers) and the list of selected principles and patterns (literals).

[2]`https://www.wikipedia.org/`, last accessed on: 2015-05-26
[3]`http://beamartian.jpl.nasa.gov/`, last accessed on: 2015-05-26
[4]`http://www.apache.org/`, last accessed on: 2015-05-26

1. *Starting communities:* Starting a new community is critical since it is the initial step in the process and errors in this phase could affect the subsequent phases negatively. The start of a new community needs to address three challenges. First, a useful niche needs to be carved out to attract members who share a certain interest. Second, this niche needs to be defended against competing communities with alternative ways users can spend their time. Third, a critical mass of users needs to generate enough content so that it can become interesting for future members. Our approach becomes more valuable once a high number of work templates for topics relevant to users are available. Most of the design principles are related to the representation of tasks that have to be matched with the right users in the work plans.

    a) Carve a niche of interest, scoped in terms of topics, members, activities and purpose

    b) Relate to competing sites, integrate content

    c) Organize content, people and activities into subspaces once there is enough activity

    d) Highlight more active tasks

    e) Inactive tasks should have expected active times

    f) Create mechanisms to match people to activities

2. *Encouraging contributions through motivation:* In our approach it is very important that users participate in the structuring of KiPs. Although the resources contributing to KiPs are often limited in organizations, motivation is still important to avoid idle time of available authors. One of the reasons could be that authors don't know what should be done or where contributions are required in KiPs. The principles in this category relate to different motivators of individual effort, e.g., individual performance, outcome and utility. This can be achieved by stressing benefits of contribution and giving small rewards that are tied to individual performance.

    a) Make it easy to see and track needed contributions

    b) Ask specific people on tasks of interest to them

    c) Simple tasks with challenging goals are easier to comply with

    d) Specify deadlines for tasks, while leaving people in control

    e) Give frequent feedback specific to the goals

    f) Requests coming from leaders lead to more contribution

    g) Stress benefits of contribution

    h) Give (small, intangible) rewards tied to performance (not just for signing up)

    i) Publicize that others have complied with requests

    j) People are more willing to contribute: 1) when group is small, 2) when committed to the group, 3) when their contributions are unique

3. *Encouraging commitment:* Commitment is important in successful communities since members that are committed work harder and stick with the community after it becomes established. Commitment is also necessary for participants to sustain the group through

problems, e.g., authors need to become familiar with the structuring concepts before they can contribute to a work template. Some important prerequisites for commitment are feelings of closeness to other individuals in the group, strong identification, or the risks of leaving the community. Most of the design principles in this category are rooted in the field theory introduced by Lewin [Le51]. In general, field theory describes the forces in people's environments that attract them to a group.

   a) Cluster members to help them identify with the community
   b) Give subgroups a name and a tagline
   c) Put subgroups in the context of a larger group
   d) Make community goals and purpose explicit
   e) Interdependent tasks increase commitment and reduce conflict

4. *Dealing with newcomers:* Online communities need to incorporate successive generations of newcomers and employees that join the organization. Newcomers are also a valuable source of innovation of new work procedures. For the goal of this thesis it is also important to effectively introduce newcomers to work templates, e.g., inexperienced knowledge workers could be guided by templates that capture best practice knowledge in an organization. Furthermore, newcomers might also cause damage in case they imperil work that has already been done by other community members on work plans. Another thread might be that newcomers always create new types because they are afraid to change already existing work templates.

   a) Members recruiting colleagues is most effective
   b) Appoint people responsible for immediate friendly interactions
   c) Introducing newcomers to members increases interactions
   d) Entry barriers for newcomers help screen for commitment
   e) When small, acknowledge each new member
   f) Advertise members particularly community leaders, include pictures
   g) Provide concrete incentives to early members
   h) Design common learning experiences for newcomers
   i) Design clear sequence of stages to newcomers
   j) Newcomers go through experiences to learn community rules
   k) Provide sandboxes for newcomers while they are learning
   l) Progressive access controls reduce harm while learning

5. *Best practices from Polymath:* Polymath allows mathematicians and volunteers that have a math background to collaboratively develop proofs for mathematical theorems [Ni12]. Volunteers in Polymath range from high-school teachers to engineers that are interested in solving mathematics conjectures. In this community they are supported with discussion threads and public blogs that are linked with wiki pages. These wiki pages are used to

write basic definitions, proof steps and the overall final publication in some situations. In case the results of a proof lead to a scientific publication, everybody who contributed to the solution of the conjecture automatically becomes a co-author. The community grows with new project proposals that are decomposed into small enough pieces of work that authors can solve. In [Mi15b], we described several patterns from the polymath community that are not covered by the aforementioned social design principles.

a) Permanent URLs for posts and comments, so others can refer to them

b) Appoint a volunteer to summarize periodically

c) Appoint a volunteer to answer questions from newcomers

d) Low barrier of entry: make it very easy to comment

e) Advance notice of tasks that are anticipated

f) Keep few tasks active at any given time, helps focus

6. *Lessons learned from ENCODE:* Another project which exposed best practices for a large collaboration is the encyclopedia of DNA elements that is presented in [Bi12]. Main goal of this research project is to build a comprehensive list of functional elements in the human genome. In this project, tasks are formally assigned to groups in the collaboration based on a fixed funding that is allocated to tasks. In [Mi15b], we described the following patterns and lessons learned.

   a) Spine of leadership with few leading scientists and 1-2 operational project managers

   b) Written and publicly accessible rules to transfer work between groups

   c) Quality inspection with visibility into intermediate steps

   d) Export of data and results, integration with existing standards

The steps that are undertaken in this thesis for the concept development with the mapping to the sections are illustrated in Figure 3.9. Structuring concepts for KiPs are developed in the first step and are presented in Section 3.2. In the second step we identify social design principles and patterns for successful online communities in Section 3.3. Based on the structuring concepts and the social design principles, we develop nine features for the structuring of KiPs in this section. We combine both steps for the design of features to ensure that knowledge workers find common ground for collaboration. In Section 3.4, the features are integrated in the user interface design of our prototype that is based on a wiki. In addition to the features for the structuring of KiPs, this section also provides additional features for the communication based on events with a social feed and a mobile interface.

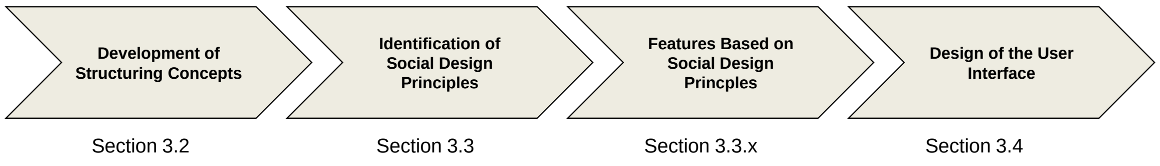

Development of Structuring Concepts
Identification of Social Design Principles
Features Based on Social Design Princples
Design of the User Interface
Section 3.2
Section 3.3
Section 3.3.x
Section 3.4

Figure 3.9.: Steps from the development of concepts to the design of the user interface

3.3.1. Task Representation

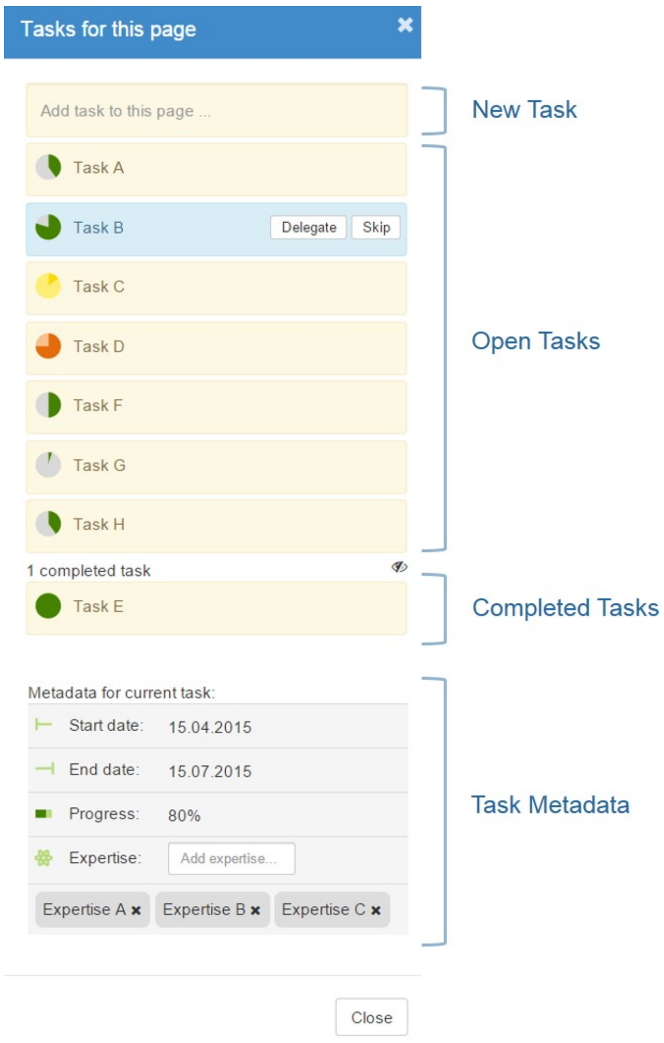
The challenge of starting a new community is mitigated based on three social design principles that the feature task representation addresses. Our representation of tasks organizes content, people and activities into subspaces (1c) based on expertises that can be assigned to every task. With these expertises subspaces of tasks that are relevant for certain knowledge workers can be grouped together. For every task it is possible to add content to the subspace through attributes. Persons can be added to a subspace using the execution roles that can also be assigned to groups. More active tasks are highlighted (1d) based on the task progress that is visualized in a small pie chart for every task. Tasks that are more active also have a higher progress visualized in the pie chart. Figure 3.10 illustrates these pie charts in the worklist of a user with several sample tasks. After selecting one task additional buttons for delegate and skip are shown in case the user has the required roles assigned. Mechanisms to match people to activities (1f) are facilitated through skills that users can collect by completing tasks with expertises assigned. Based on these skills suitable users for tasks can be determined.

Several social design principles for the task representation are applied to encourage contributions through motivation. With the pie charts for tasks it is easy to see and track needed contributions (2a). Tasks that need more contributions have a lower progress, e.g., task G in Figure 3.10 has the lowest progress of all tasks. This is not only visible for simple tasks but also for entire wiki pages since it is possible to compute the average progress for all tasks assigned to a wiki page as described in Section 3.2.3. Complex tasks can be incrementally broken down to simple tasks (2c) so that they are easier to comply with. Another advantage of fine-grained tasks is that they can be assigned to small groups which are more willing to contribute. In addition to smalls groups tasks can also be divided for groups that have many committed members to encourage motivation. Another advantage of the task representation is that contributions become unique and completed tasks remain in the worklist of a wiki page (2j), e.g., task E that is completed remains visible at the bottom of the worklist. Completed tasks are per default hidden to avoid confusion and can be shown using the eye button.

Four social design principles are integrated in the representation of tasks to encourage commitment. Subgroups that work together get an own name for their task, i.e., instead of task A to H it is possible to specify an arbitrary name (3b). Subgroups are always part of a larger group context since there are several tasks assigned to a wiki page in the most cases, e.g., the subgroup that is responsible for task B is in the context of the remaining tasks that are assigned to other subgroups (3c). Our task representation allows it to make community goals and purpose explicit (3d). Every goal can be described as high-level task with an unlimited number of subtasks. Finally, the task representations are interdependent to increase commitment (3e). The progress of every task on a wiki page is taken into account for the overall progress of the wiki page, i.e., the progress of the wiki page can only be 100% after all tasks on this page are completed. The task representation feature incorporates two patterns observed in Polymath and one pattern from ENCODE. Tasks can be represented with permanent URLs so that others can easily refer to them (5a). Notice of tasks that are anticipated are advanced with an empty pie chart in the worklist (5e). Tasks that are anticipated can be created in the system without any metadata about their start and end date. Completed tasks are visualized with a green pie chart similar to the quality inspection in ENCODE (6c).

3.3.2. Task Metadata

Metadata for the selected task are shown at the bottom of Figure 3.10 in the gray box. More active tasks can be highlighted (1d) in the task representation with the specified progress in the metadata. The progress can be changed by users manually to estimate how much of the task is already completed. Once the progress is changed the corresponding pie chart in the task representation is automatically updated. Start and end dates in the metadata can also be used to specify that currently inactive tasks are going to have expected active times in the future (1e), i.e., the start date of the inactive task is not yet reached. The last value in the task metadata is used to specify expertises that describe what kind of skills are required to complete the task (1f), e.g., expertises A, B and C. Similar to the highlighting of more active tasks, needed contributions can be marked using the progress in the metadata (2a). Specific people can be asked on tasks of interest to them using the expertises that can be assigned to tasks in the metadata (2b).


Tasks for this page

Add task to this page ...

Task A

Task B Delegate Skip

Task C

Task D

Task F

Task G

Task H

1 completed task

Task E

Metadata for current task:

Start date:	15.04.2015
End date:	15.07.2015
Progress:	80%
Expertise:	Add expertise...

Expertise A ✖ Expertise B ✖ Expertise C ✖

Close

Figure 3.10.: Worklist with sample tasks

Deadlines for tasks can be specified using the end date for tasks in the metadata (2d). Users are at any time in control of the deadlines and main purpose of them is to remind users when certain tasks are due. Thereby, users are not forced to complete overdue tasks but they remain in control of the course of action at any time. Tasks that are not completed after the specified deadline are highlighted with a red pie chart in the worklist, e.g., task D in the sample worklist. A detailed description of the task state feature is presented in Section 3.3.4. The social design principle to give frequent feedback to specific goals is also supported with the task metadata (2e). Users working on a specific task can update the progress to provide feedback every time they have some advances with the goals of the task. Another possibility to encourage contribution through motivation is based on the social design principle (2f). According to this principle requests coming from leaders lead to more contributions of users. This principle can be supported using the delegate role, i.e., leaders can delegate tasks to other knowledge workers to encourage motivation.

The challenge encouraging commitment is addressed with three social design principles in the task metadata feature. Members are clustered to help them identify with the community (3a) based on skills. Users are gaining skills in their profile every time they finish a task that has expertises assigned, e.g., the user finishing the task B from Figure 3.10 gains the expertises A, B and C in his profile. In addition to the name of a task, expertises in the metadata can be used to give subgroups a tagline (3b). Interdependence of tasks can be supported through the metadata with start and end dates that come directly after another task (3e), i.e., in order to start one task another task needs to be finished first. This interdependence increases the commitment since users notice that other tasks depend on their results. Quality inspection with visibility into intermediate steps (6c) is supported with the progress that is only set to 100% when the goals of the task are achieved. Intermediate steps are visible using the subtask navigation that is described in Section 3.3.6.

3.3.3. Task Management

The task management allows to organize content, people and activities into subspaces once there is enough activity (1c). These subspaces are created by adding new tasks to pages which is illustrated at the top of Figure 3.10. Users only have to specify a name and confirm with enter to add a new task. The input field and tasks are not visible for visitors in the worklist since they don't have edit rights. Content for the task can be assigned using drag & drop functionality in the user interface, i.e., attributes can be dragged from the wiki page and dropped on the task. Tasks with attributes are automatically completed once all constraints on the assigned attributes are fulfilled. In case no attributes are assigned to a task, the completion depends on the manually updated progress in the metadata. To create subtasks for a task t attributes that reference other wiki pages with tasks have to be assigned to t. This feature can be used to create simple tasks that are easier to comply with (2c). With the creation of tasks and subtasks the social design principles (2j) can be fulfilled. Subtasks can reduce the size of the group in case too many users are involved to make sure that people are more willing to contribute. The task management allows the creation of intermediate steps (6c) for quality inspection, e.g., task a is completed based on the intermediate steps *a1*, *a2* and *a3*.

3.3.4. Task State

The state of tasks is automatically determined based on the task metadata introduced in Section 3.3.2. Main purpose of the task state is to make it easy to see and track needed contributions for users (2a). The task's state is visualized in a small pie chart that is shown next to every task in the worklist and presented in [Gi15c]. An overview of possible task states depending on the metadata of tasks is illustrated in Figure 3.11. States can be either active or fade out in case they are only used in the context of an active task. Immediately after creation of a task the state is displayed with the pie chart in the first row. This allows to advance notice of tasks that are anticipated although exact dates cannot be specified (5e). The circles are empty to illustrate the incomplete metadata of tasks since it is not possible to compute the progress. Once the metadata is in progress the circle is filled depending on the number of completed values. The metadata is complete as soon as start and end dates are specified, i.e., it is not necessary to enter expertises.

Tasks with complete metadata are visualized with filled pie charts, whereas the computation of the progress is described in Section 3.2.3 and used to determine the fill levels of pie charts. Overdue tasks are represented with a red pie chart independent of the current progress. In case tasks contain one or more overdue subtasks, this is indicated with a small circle at the bottom of the pie chart. Tasks can also be in an inconsistent state in case the start or end dates exceed at least one parent task, e.g., parent task t ends on 01st January and subtask s of t ends on 15th January. Inconsistent tasks are represented with a yellow pie chart or an empty yellow circle. Tasks that contain inconsistent subtasks are represented with a small yellow triangle at the bottom of the pie chart. All these task states are used to provide frequent feedback to users (2e) and make the current state of tasks transparent.

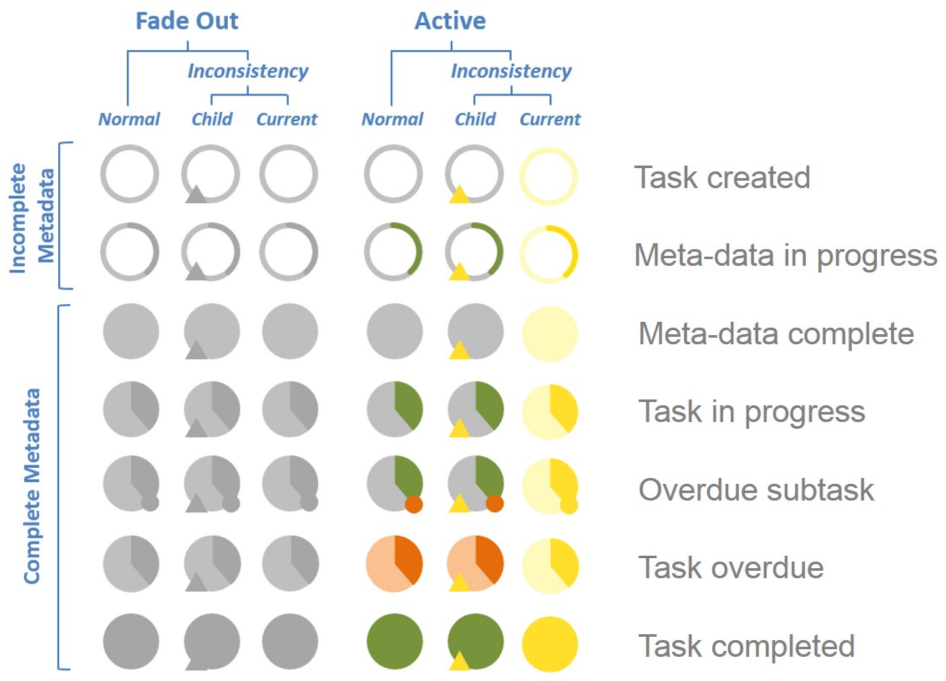


Figure 3.11.: Possible task states depending on their metadata [Gi15c]

The most likely state transition sequences between these tasks are illustrated in Figure 3.12. At the top of the figure the normal state transitions are displayed in two phases. In the first phase, metadata are incomplete and entered by users with dates and required expertises for the task that are optional. The progress of entering metadata is visualized by the progress of the surrounding circle. This phase is finished with completed metadata and the resulting filled out pie chart is gray since there is no progress related to the goals of the task at this moment of time. This changes once users start working on a task with the progress increasing in green color as long as the specified due date for the deadline has not passed. Within the subtasks several tasks might become overdue and users are reminded with a small red circle, so that the overdue subtasks tasks that are highlighted with red pie charts are completed. This sequence finishes with a green pie chart indicating that all subtasks of this task are completed and nothing needs to be done.

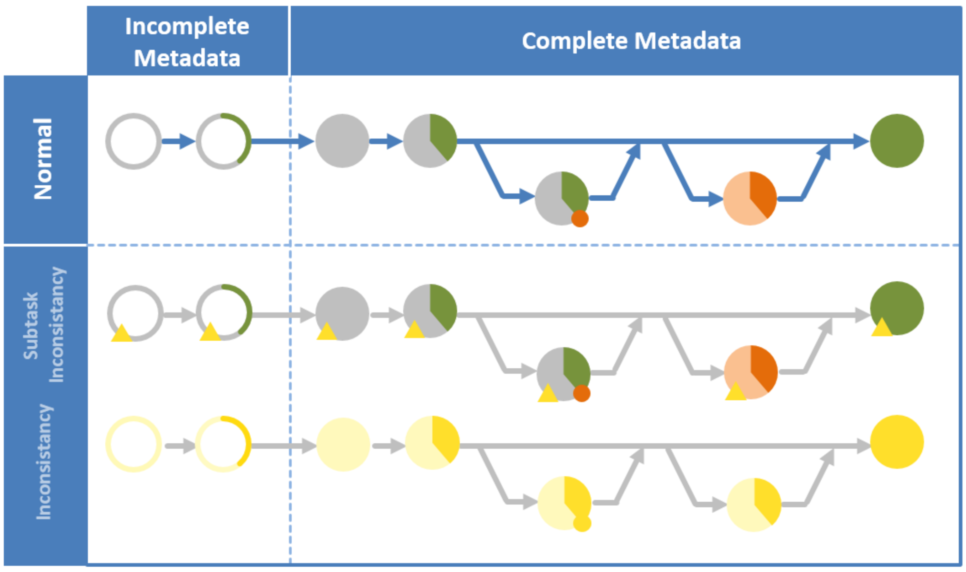


Figure 3.12.: Most likely state transition sequences of tasks [Gi15c]

At the bottom of Figure 3.12 two additional sequences are illustrated with inconsistent tasks and subtasks. The first sequence starts with a small yellow triangle in the first phase indicating at least one inconsistent subtask within the hierarchy. This is a common scenario in case existing tasks are assigned to the new task that has no dates specified in the beginning. Similar to the normal transition sequence presented at the top of Figure 3.12, the subsequent states have a higher progress until the task is completed. While the completion of tasks is not possible as long as overdue subtasks exist, inconsistent subtasks have no influence on the overall progress, i.e., tasks containing subtask inconsistencies can be completed. One possible state transition with an inconsistent task is shown at the bottom of the figure. Similar to the previous transitions the inconsistent task passes through the typical states. This is possible despite the fact that the start and end dates are inconsistent for this task. Similar to attribute constraints that can be violated as long as they are not defined as strict, we allow inconsistencies for tasks that have wrong start and end dates so that users are not limited in their flexibility. The state transition sequences of tasks are only illustrated with three likely examples, but many other combinations of state transitions are conceivable.

3.3.5. Timeline Navigation

The timeline is automatically generated based on the provided metadata of tasks. Moreover, it helps to highlight more active tasks (1d). Figure 3.13 shows a screenshot of the timeline based on the previously introduced sample tasks. The timeline only shows open tasks since completed tasks are automatically removed. The length of the bars is determined based on the duration of tasks that is computed from the start and end dates, i.e., tasks with a higher duration have longer bars. Goal of the timeline is to provide a simple overview about the process, so that we decided not to involve too many details. The red vertical line shows the current date which makes it very easy to see and track needed contributions (2a), i.e., task A is slightly behind the schedule and requires some contributions. The timeline also visualizes when tasks have scheduled active time (1e), e.g., task C will have some active time soon. Tasks that have incomplete metadata are shown with an empty bar to advance notice of tasks that are anticipated (5e), i.e., these tasks are visible in the timeline even without specified metadata to inform users of anticipated tasks in the future.

The colors indicating the state of the tasks is used consistently with the pie charts that are shown in the worklist. Task C is yellow since it inconsistent with the parent task and task D is red because it is only at 75% although the deadline was already reached in the past. The page progress shown as pie chart next to the title represents the average progress of all tasks attached to this page (6c), i.e., the test page has an overall progress of 51%. In addition, two small icons are shown at the bottom of the pie chart to indicate the inconsistent and overdue tasks. The timeline only shows tasks at the same hierarchy level like the current wiki page, i.e., subtasks are not visible in the timeline as long as no task is selected. The subtask navigation feature based on the timeline is described in Section 3.3.6. Main purpose of the timeline is to make the progress transparent and required contributions visible for the users. Therefore, it is not possible to edit any data or metadata within the timeline.

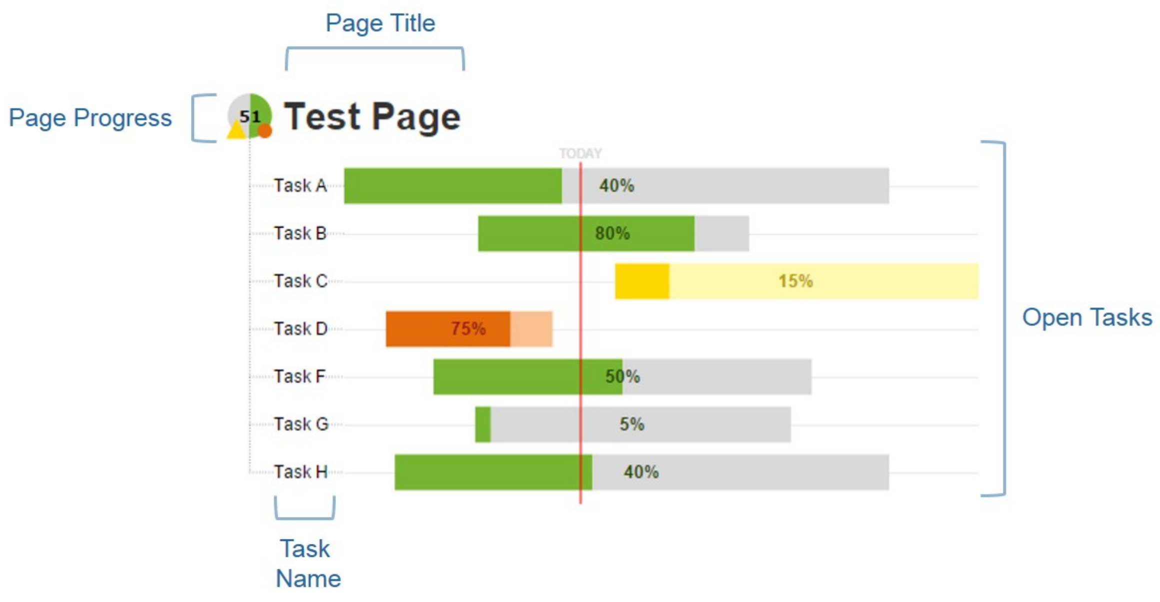


Figure 3.13.: Timeline visualizing the progress of open tasks

3.3.6. Subtask Navigation

The subtask navigation feature is a combination of the task representation in the worklist (Section 3.3.1) and the timeline navigation (Section 3.3.5). It allows users to quickly get an overview about the progress of subtasks s that belong to another parent task t. By selecting task t in the worklist all subtasks s are represented in the timeline feature with their progress and the assigned attributes with references to wiki pages containing these subtasks are filtered. In case t has no subtasks the timeline is shown as empty without any tasks. This might be the case when only attributes with a simple type or no attributes at all are assigned to t, i.e., no navigation to subtasks is required. Main advantage of the subtask navigation is that it is not necessary for users to change to the lower level wiki page, which avoids time consuming page reloads in the browser. The subtask navigation can be used to traverse through the hierarchical task structure to identify needed contributions, e.g., tasks that have a low progress in the timeline. In case a subtask that requires attention has been identified, a user can select the attribute assigned to the parent task in order to navigate to the wiki page with this subtask. This step can be repeated at will to traverse to tasks that are on lower levels in the hierarchy.

This feature for the subtask navigation encourages contributions through motivation with three social design principles that are mentioned and briefly explained in the following. First, it becomes easier to see and track needed contributions (2a) with the subtask navigation. Users can easily navigate to subtasks of open tasks to identify missing contributions that prevent the parent task from having a higher overall progress. Second, it also supports frequent feedback to specific goals (2e) since every contribution on the particular lower level in the hierarchy becomes immediately visible. In collaborative environments subtasks of a wiki page p might be assigned to different users and groups, i.e., the subtasks have other users and groups than the parent tasks that are assigned to p. Third, with the subtask navigation feature it becomes public that others have complied with requests (2i). In this context, tasks can be considered as requests to others. Once the responsible users of the parent task can see contributions on related subtasks, they will be encouraged to contribute more on their own tasks as well according to this social design principle.

With the subtask navigation feature another social design principle is considered that encourages commitment in the community. This principle relates to interdependent tasks that increase commitment and reduce conflict (3e). Users of a task are more committed since they can see subtasks that are related to them. Subtasks are interdependent since they influence the progress of the parent task. While the task representation feature introduced in Section 3.3.1 supports this principles for tasks on the same hierarchy level, the subtask navigation feature achieves this for tasks on a lower level in the hierarchy. Conflicts are reduced because interdependent subtasks are made explicit and users are aware of potential conflicts. Finally, the social design pattern on quality inspection with visibility into intermediate steps is supported through the subtask navigation feature (6c). Subtasks can be considered as intermediate steps of a parent tasks. With the subtask navigation users can evaluate whether important intermediate steps are undertaken and finished to assure quality of the information objects created by the parent task.

3.3.7. Task Alert

Knowledge workers usually operate in different context and they have to be aware of many tasks at the same time [Mu12]. The task alert feature remembers knowledge workers about overdue tasks. Tasks become overdue in case the specified end date has passed and the progress is still lower than 100%. Without this feature users would have to use the subtask and timeline navigation to identify overdue tasks. Figure 3.14 illustrates the alert feature with the sample overdue task D, which should have been finished 13 days ago and currently only has a progress of 75%. Overdue tasks in the alert are sorted by their end date, i.e., tasks that have the furthermost end dates are shown at the top. It is not possible to hide overdue tasks so that users are always reminded about them. There are two possibilities to remove overdue tasks from the alert. First, the task is completed by the user on the wiki page. Second, the end date of the task is changed to a later point in time in the future. The alert feature contains overdue tasks that are assigned to the user from arbitrary pages and wikis. Users can navigate to overdue tasks by clicking on the link to the page that is shown below the task name.

The task alert feature is designed based on two social design principles to encourage contributions through motivation. First, needed contributions are visible with the progress of tasks, i.e., uncompleted tasks with a progress lower than 100%. In a similar way overdue tasks require contributions that have a higher priority since the end dates are already passed. The task alert feature makes it easy to see and track these needed contributions (2a). According the second social design principle deadlines have to be specified for tasks, while leaving people in control (2d). End dates for tasks in our approach can be considered as deadlines since they are used to specify when a task needs to be finished. The users in our approach are at any time in control of the deadlines, i.e., they can change the end date of their tasks at any time. Although this might sound contradictory at the first glance, it is very important that users are able change the start and end dates of their tasks. As described in Section 2.2, knowledge workers require a high degree of autonomy and self-organization to fulfill their tasks due to the unpredictable characteristic of knowledge work. Despite this flexibility for knowledge workers it is still possible to describe fixed end dates, e.g., deadlines for project deliverables that cannot be delayed. For this purpose parent tasks can be defined with fixed end dates, e.g., by project managers. Thereby, knowledge workers are still able to change the end dates of their tasks, but a warning in the parent task is automatically created (cf. task states in Section 3.3.4). The responsible users of the parent and subtasks need to resolve the conflicting tasks collaboratively.

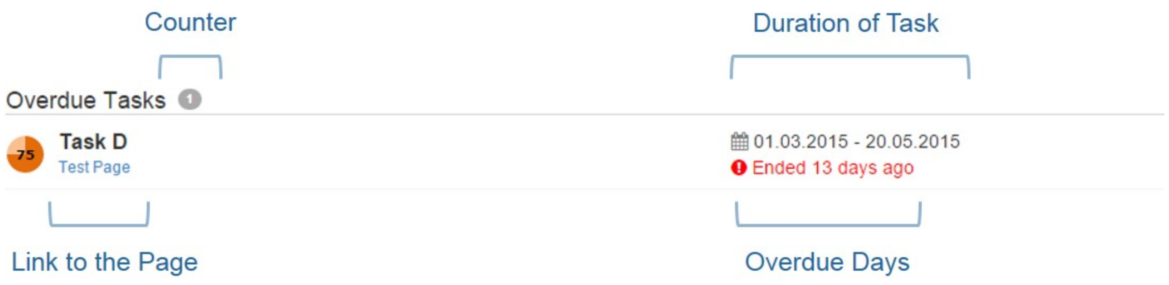


Figure 3.14.: Alert feature visualizing open tasks that have passed the specified end date

3.3.8. User Rating and Skills

One of the requirements for software support of KiPs that is described in Section 2.2 is concerned with the modeling of knowledge workers' skills and resources. Figure 3.15 shows a screenshot of the user rating and skills feature. The skills visualization shows the distribution of the most important expertises for the user. To illustrate this feature we completed one sample task with this user account. In this example, the completed task has three expertises assigned, i.e., the expertises in the skills visualization are equally distributed. By hovering over an expertise the total number of completed tasks with this expertise and the relative share as percentage of this expertise are displayed. The visualization only shows the expertises having at least 5% share and never contains more than ten expertises for readability reasons. The ranking visualizes the percentile of completed tasks compared to the remaining users in the system. The computation of the ranking might also be weighted depending on the duration or difficulty of the task, e.g., some expertises might have a higher multiplier. New tasks could be assigned to users based on their skills and their rating in the system.

This feature encourages contribution through motivation and commitment since it is based on several social design principles. Specific users can be asked on tasks of interest to them (2b), whereas we consider tasks that require expertises that a users has already collected as interesting. Users also receive frequent feedback about their personal goals (2e), e.g., in case they want to gather certain expertises the skills visualization is updated every time they finish a task. The skills and ranking gives users rewards in the profile that are tied to their performance (2h). According to social design principle (2j) users are more willing to contribute when the group is small, they are committed to the group and contributions are unique. These groups and contributions can be organized around certain expertises. Commitment can be encouraged by clustering members based on their expertises (3a), so that they can identify with the community, e.g., clustering all users having the expertise A. Finally, users might be more committed to tasks that are related with expertises that belong to their skills (3e).

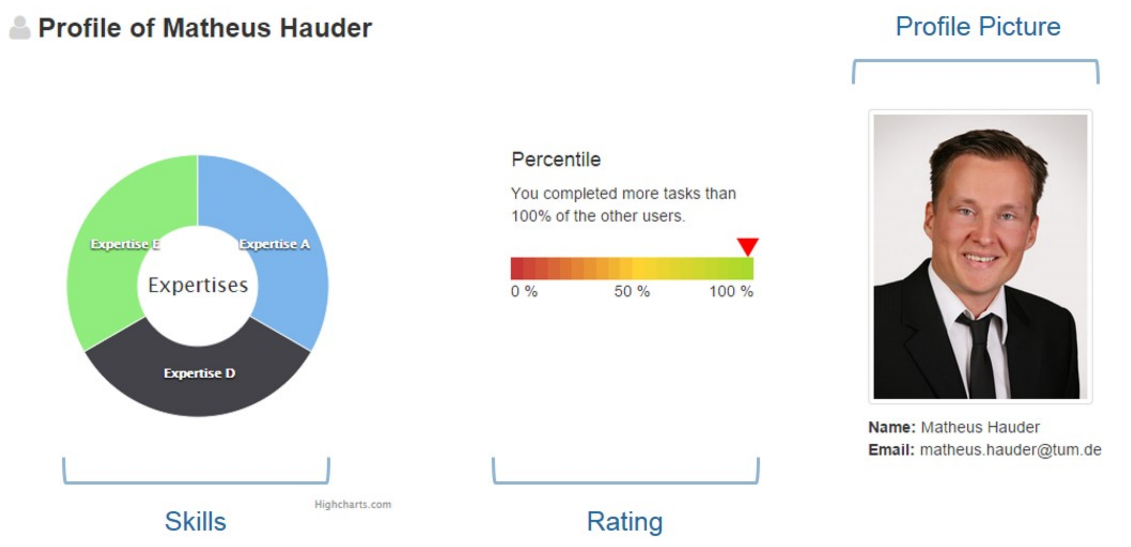
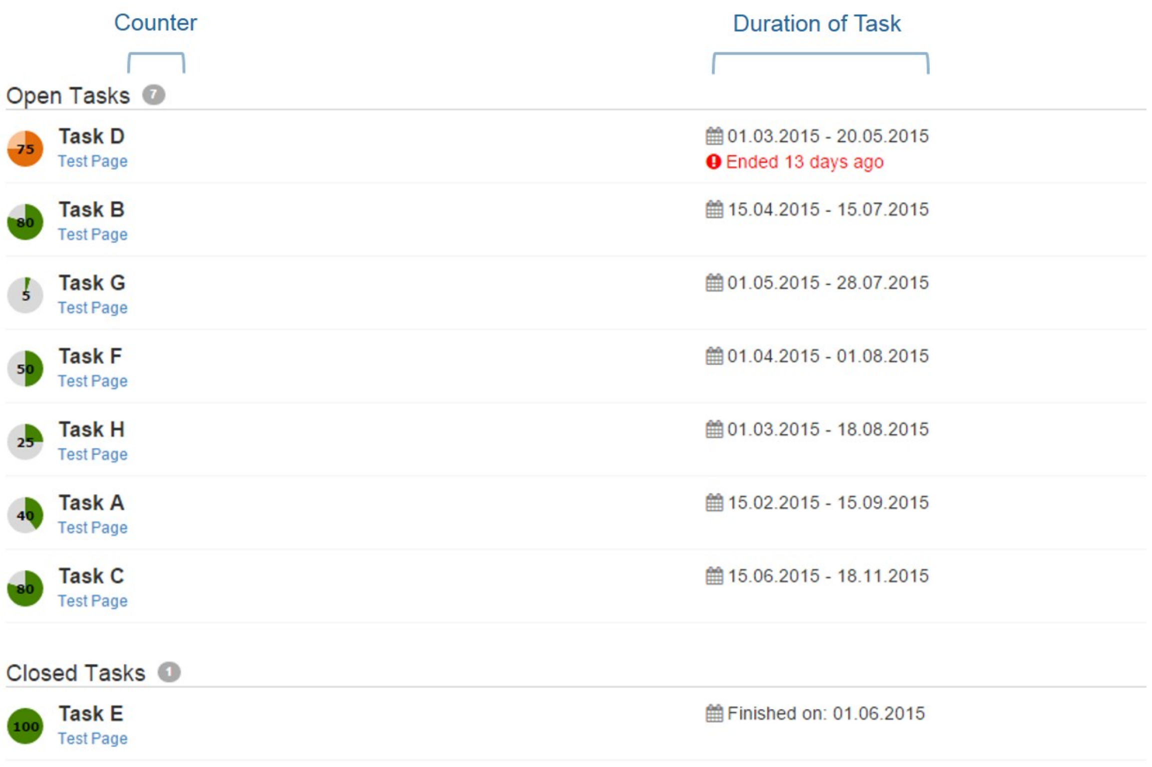


Figure 3.15.: User profile of the author after completion of a task with sample expertises

3.3.9. Personal Worklist

Every user has an own personal worklist that summarizes all tasks assigned to this user. Figure 3.16 shows a screenshot of the personal worklist with the previously created sample tasks that are assigned to a user. The worklist is divided into two sections for open and closed tasks. Main purpose of the personal worklist is that users see all tasks that are assigned to them. In our examples, we assigned tasks only to one wiki page for readability reasons. Usually tasks assigned to a user will be located at many different wiki pages. The personal worklist is useful for its owner to prioritize open tasks for processing. Tasks are ordered by the end date in the worklist, i.e., the next due task is always shown at the beginning. Other users can take a look in the personal worklist to get an overview about what colleagues are currently working on.

The social design principle to highlight more active tasks (1d) is considered in this feature, because only open tasks ordered by their due date are shown in the personal worklist. In case a user has overdue tasks assigned, they are also shown at the top of the personal worklist. Every user has an overview about his required contributions (2a), i.e., all tasks assigned to a user are visible in his personal worklist. Users are in control of their deadlines and the personal worklist helps them to specify these deadlines based on their future workload (2d). Without an overview about future obligations it is hardly feasible to schedule deadlines for tasks. Based on their obligations that are shown in the personal worklist, users can manage deadlines for their upcoming tasks. For example task F is scheduled shortly after task G, although the progress of task G is just at 5% indicating that much work is still outstanding.



Open Tasks 7

75	Task D Test Page	01.03.2015 - 20.05.2015 Ended 13 days ago
80	Task B Test Page	15.04.2015 - 15.07.2015
5	Task G Test Page	01.05.2015 - 28.07.2015
50	Task F Test Page	01.04.2015 - 01.08.2015
25	Task H Test Page	01.03.2015 - 18.08.2015
40	Task A Test Page	15.02.2015 - 15.09.2015
80	Task C Test Page	15.06.2015 - 18.11.2015

Closed Tasks 1

100	Task E Test Page	Finished on: 01.06.2015

Figure 3.16.: Personal worklist showing all sample tasks that are created for the user

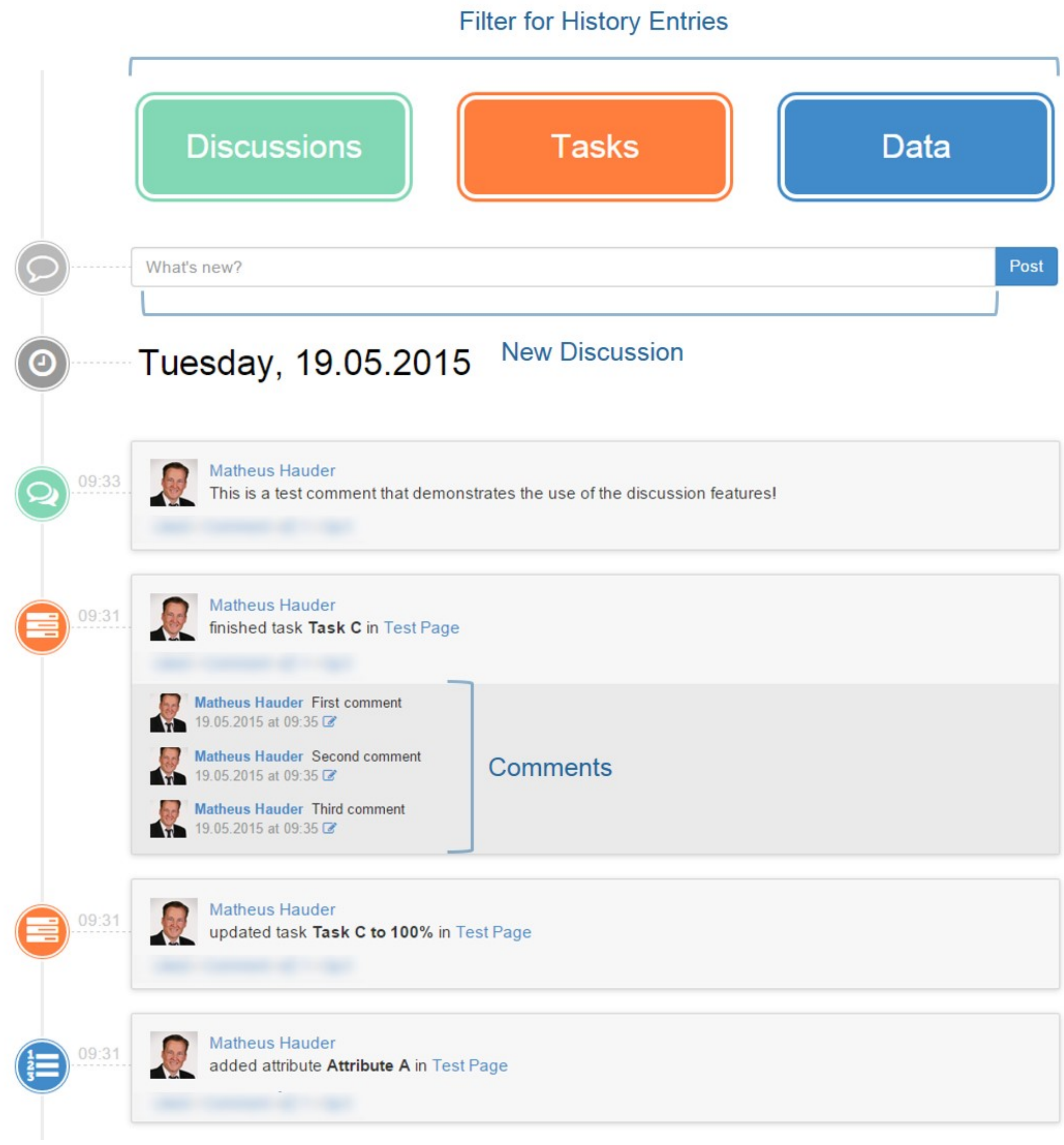
3.4. Design of the User Interface

This section presents he design of the user interface based on the features introduced in Section 3.3. The features and the design of the user interface are presented in two separate sections since the features are quite independent of the user interface design. Depending on the application scenario the proposed design might be adapted without changing the set of features, i.e., the task representation could be shown more prominently in the user interface. Other possibilities could be the design of domain-specific user interfaces or the integration of the features in existing applications. In Section 3.4.1, the user interface of the social feed is introduced that is shown as landing page in the system. In Section 3.4.2, the representation of work plans on wiki pages is described, whereas work plans are an extension of Hybrid Wiki pages introduced in [Ne12]. Finally, Section 3.4.3 introduces the user interface design of the mobile website. Due to the smaller screen size on mobile devices the features for KiPs and possible user interactions have to be adapted, i.e., the set of user interactions on the mobile client is reduced. The design introduced in this example should not be treated as fixed for all application scenarios of KiPs and it is rather a suggestion for a possible implementation of our approach.

3.4.1. Social Feed

Although the social feed is not directly related to the structuring of KiPs, it is an important part of the user interface. It improves communication among knowledge workers in the context of history events. The history events in the social feed are the same as illustrated in Figure 2.6 within the introduction of the thesis. Three kinds of different history events are distinguished in the social feed. (1) *Discussion* history entries are simple text strings that are posted by users in the system. (2) *Task* history entries are automatically generated after tasks changed their progress or were completed. (3) *Data* history entries are automatically generated after attributes on work plans are changed. Additional history events are currently not considered since workers should only receive the most important events that are related to their work. Figure 3.17 shows an annotated screenshot of the social feed that is initialized with the same sample data like in the previous examples for the features. The social feed is shown as landing page in the system for every user, i.e., it is shown always after users successfully logged in to the system. New history entries are highlighted for every user individually with a red badge that contains a counter in the navigation bar. This counter is reset after the user has seen the social feed.

Main advantage of the social feed is the possibility for knowledge workers to communicate within the context of a history event, e.g., after someone completed a task other workers can comment this event (cf. comments in Figure 3.17). Commenting is possible for all three kinds of history events in the social feed and the comments are updated in real-time to facilitate discussions about history entries. The three buttons shown at the top of the social feed can be used as filters, i.e., after clicking on one or more buttons the respective history events are filtered. Every history event is associated with one color that is used consistently throughout the user interface. Another advantage of the social feed is that workers are aware of all activities that are performed within the KiPs. In particular for highly collaborative settings this can be a very valuable feature since it might avoid time consuming meetings between workers that are responsible for the same or interrelated tasks. Existing activity feeds that can be found in many Enterprise 2.0 applications are quite unstructured, e.g., the activity feed used for Hybrid Wikis [Ne12]. The social feed presented in this thesis takes advantage of the structure that is based on the lightweight structuring concepts provided by authors, i.e., task C was updated to 100% (cf. Figure 3.17). Without this structure the information about the progress of tasks would only be available implicitly in unstructured activity feeds. In the prototype history entries are never deleted from the social feed and users can go back the entire history. In case a huge number of events are generated, workers might receive events in their social feed that are not directly relevant for them. In future work intelligent algorithms could be applied to prioritize history events for every user.



Tuesday, 19.05.2015

09:33
Matheus Hauder
This is a test comment that demonstrates the use of the discussion features!

09:31
Matheus Hauder
finished task **Task C** in Test Page

Matheus Hauder First comment
19.05.2015 at 09:35

Matheus Hauder Second comment
19.05.2015 at 09:35

Matheus Hauder Third comment
19.05.2015 at 09:35

09:31
Matheus Hauder
updated task **Task C to 100%** in Test Page

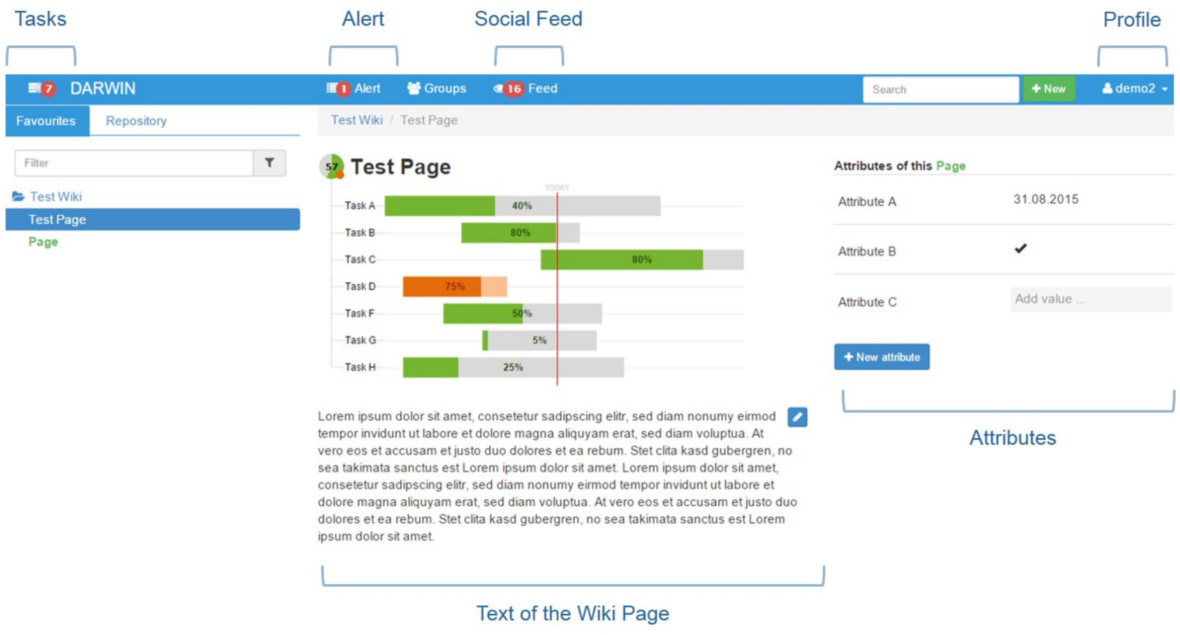
09:31
Matheus Hauder
added attribute **Attribute A** in Test Page

Figure 3.17.: Social feed with activities that are performed in the work plans

3.4.2. Representation of Work Plans

Figure 3.18 shows a screenshot of a work plan that is based on the same sample data that is used in the previous examples. The timeline feature is placed prominently at the top of the wiki page to lead the attention of the user on the progress of current tasks. Below the timeline the unstructured text of the wiki page is shown, which can be edited by clicking on the pencil button. On the right hand side of the wiki page the attributes of the *Test Page* are shown. In this example, the three attributes have different data types. *Attribute A* contains a date attribute value whereas the date is entered with a date picker. *Attribute B* contains a boolean value that is entered with a simple checkbox and *Attribute C* is a string value. New attributes can be added with the *New Attribute* button that is shown at the bottom of the infobox. On the left hand side users can navigate to other wikis and wiki pages with the explorer menus. Work templates are written in green color at the bottom of every wiki, e.g., the work template *Page* that is used in the example.

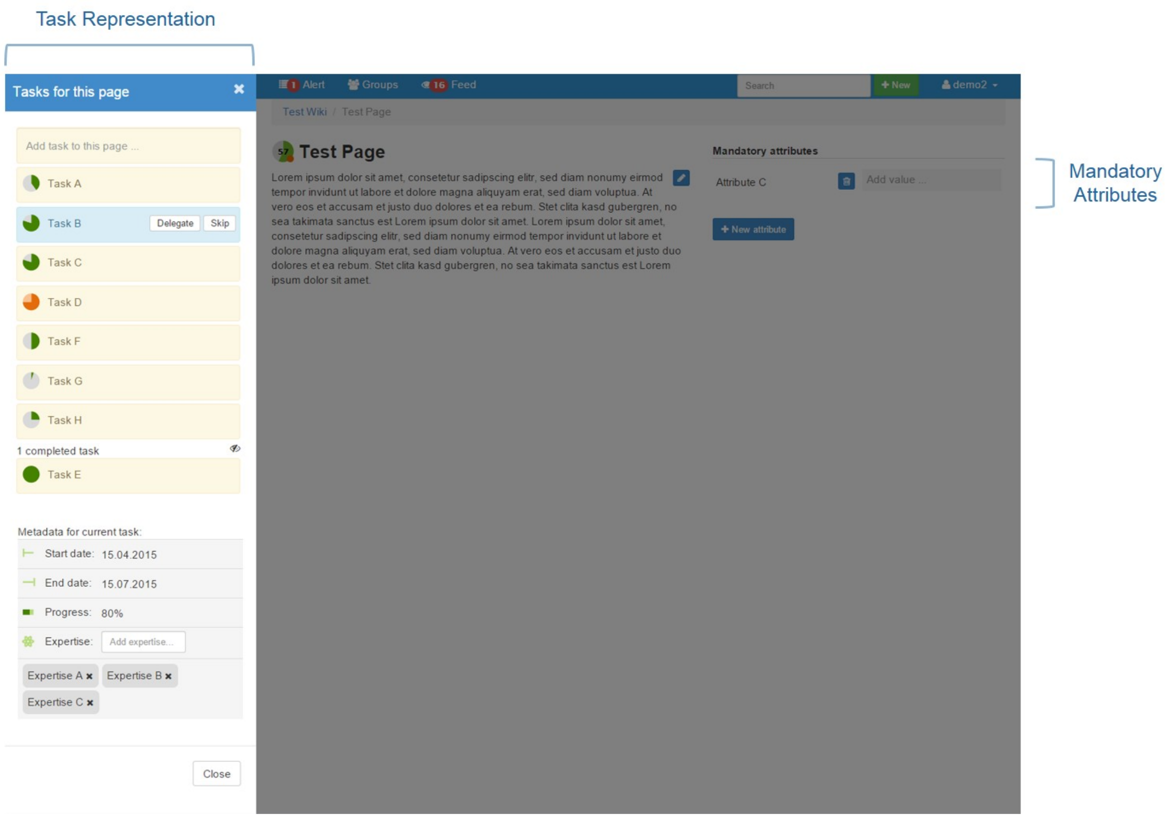
The navigation bar is located at the top of the screenshot and always visible on every page. The side window for the task representation can be enabled on the left hand side of the navigation bar. The counter in the red badge indicates how many tasks on the current wiki page are enabled. The *DARWIN* logo can be used to navigate back to the social feed. All overdue tasks for the user are shown with the alert feature that also contains a red badge with a counter. Next to the alert feature existing groups are shown in the system. The counter for the social feed shows how many new history entries have been generated since the last visit of the feed. Next to the full text search a green button is provided to create new wikis and wiki pages. During the creation of new wiki pages an existing or new type can be assigned. Finally, the name of the user is shown next to the new button that can be used to enter the profile page. The arrow next to the profile opens a context menu to logout from the system.



Attribute A	31.08.2015
Attribute B	✓
Attribute C	Add value ...

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Figure 3.18.: Screenshot of a sample work plan that is represented on a wiki page for authors



Test Wiki / Test Page

Test Page

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Mandatory attributes

Attribute C: Add value ...

1 completed task

Metadata for current task:

Start date: 15.04.2015
End date: 15.07.2015
Progress: 80%
Expertise: Add expertise...
Expertise A, Expertise B, Expertise C

Figure 3.19.: Screenshot of the sample work plan with an enabled side window for tasks

Figure 3.19 shows a screenshot of the same sample page with an enabled side window for tasks. The side window can be enabled by clicking on the task icon in the navigation bar or by dragging an attribute. After the side window is enabled the remaining page is immediately grayed out and the width of this window is the same like the explorer for the navigation. With an enabled side window it is still possible to read the entire wiki page. After a task is enabled all mandatory attributes assigned to the task are filtered (cf. Task B and Attribute C in Figure 3.19). The side window is closed as soon as the user clicks into the gray area of the wiki page or uses the close button. After the side window is closed the task remains activated so that users can enter the mandatory attributes to complete the enabled task. Next to the filtering of mandatory attributes the timeline of subtasks is shown in case the enabled task has attributes assigned that reference wiki pages with tasks.

If the selected task has attributes with pages as values assigned, the timeline is shown with tasks of these subpages so that users can easily browse through subtasks. The filtered mandatory attributes can also be used to navigate to subpages. Tasks can be completed by setting the progress to 100% in case no attributes are assigned. Nevertheless, user are not forced to enter missing attribute values in this way. Values for attributes can be entered directly on the wiki page without enabling a task before. In this case the progress of related tasks is updated by the system in the same way. In our approach we consider tasks as means to guide knowledge workers with the steps that they might undertake. However, tailors might specify dependencies between tasks with rules to enforce certain traces if this is necessary. Sometimes this might be necessary since it is not always desired to give authors absolute freedom.

3.4.3. Mobile User Interface

The design of the mobile user interface is illustrated in Figure 3.20 with three screenshots. A detailed description of the mobile user interface is available in the master's thesis in [Ab15]. Due to the lower screen size some pages have to be separated with tabs, e.g., the profile has three tabs for closed tasks, open tasks and expertises. The mobile interface has a reduced set of features for tasks and attributes since some use cases are less relevant. It is not possible to assign attributes to tasks with the drag and drop feature that is available in the desktop version. Another limitation is that there is no possibility to upload files for attributes. These features are very difficult to implement in a satisfying way and less frequently used on a mobile device. The majority of the other features is executable in the mobile interface, e.g., creating tasks, wiki pages, attributes and wiki pages.

The social feed is very similar to the desktop version with three filters at the top for discussions, tasks and data. Users can comment every history entries in the feed. Due to less relevance the features for editing comments and the overview of users that liked a history entry are disregarded. One extension is the infinite scrolling that automatically reloads feed entries. Wiki pages are divided into three tabs for tasks, text and attributes. Creating and editing is possible for tasks and wiki page text. Features that are disregarded are the editing of attributes, timeline and assignment of attributes to tasks. The selection of tasks with the filtering of mandatory attributes is available for authors. The editor for the wiki page text is simplified to avoid the toolbar on the small screen. In the profile page expertises are summarized with bars, whereas it is not possible to filter tasks by selecting individual expertises.

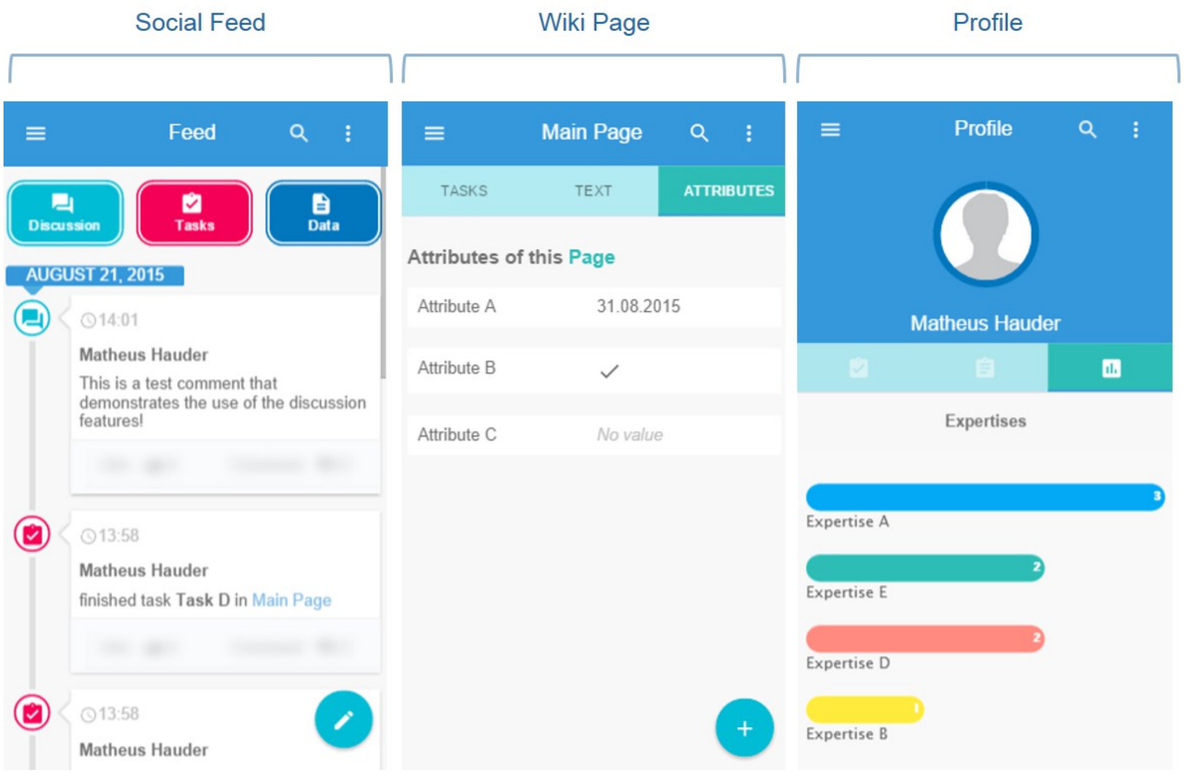


Figure 3.20.: Screenshots of the social feed, wiki page and profile in the mobile user interface

Workers who leverage knowledge and make decisions about the course of actions account for 25% to 50% of the workforce.

Tom Davenport

CHAPTER 4

Implementation

The concepts for structuring KiPs presented in Section 3 are implemented in a software solution called Darwin. This software solution is used for the case studies and the evaluation that are presented in the subsequent Chapters 5 and 6. In the following three sections of this chapter implementation aspects of the prototype are presented in detail. In Section 4.1, we present the implementation related to the lightweight structuring concepts for end-users. This includes an overview about the architecture and used frameworks which provide the technological foundation. The implementation of the conceptual data model and the Application Programming Interface (API) of the server are described. Finally, some selected implementation aspects that are relevant for authors are presented. In Section 4.2, the workbench for CMMN that is used by tailors is introduced. After the underlying technology is explained, the method complexity based on our specification subset is computed. The workbench implementation is explained based on the interaction with elements of the process. In Section 4.3, some implementation aspects that are relevant for the mobile interface are described. Similar to the previous sections the underlying technology is explained in a first step. Among the selected implementation aspects for the mobile interface is the navigation. This is a challenging issue due to our extensive metamodel that captures process as well as data structures. This chapter concludes with the implementation of the infinite scrolling for the social feed that is implemented in the mobile interface.

4.1. Software Support for Knowledge-Intensive Processes

This section describes the software solution that is developed within this thesis to demonstrate and evaluate the feasibility of our approach to support the collaborative structuring of KiPs. The software solution is developed from the scratch although it extends the concept of Hybrid Wikis [Ne12]. The initial design of the software architecture and a basic implementation is provided in [Bl13]. Main goal of the development is a lean implementation that can be used to evaluate our research questions introduced in Section 1.2. Therefore, unimportant features that are not related to the evaluation of these research questions are not covered in the current version of the implementation. In order to use the solution in an organization several extensions might be necessary, e.g., versioning of wiki pages, integration with directory services and an improved management of user accounts with an integration to other systems. However, the solution is designed to be easily extensible and scalable so that an eventual extension with these capabilities is possible.

4.1.1. Architecture

The architecture of the implemented software solution is illustrated in Figure 4.1. Starting from the top with the participating roles in our approach, every role has on own user interface that is designed specifically for their purpose. All user interfaces are developed for web browsers which makes them platform independent. The generic user interface allows authors to structure KiPs with the work plans as described in Section 3.4.2. In addition, authors are also able to use the software from a mobile device with the interface presented in Section 3.4.3. Visitors can also use both interfaces but they are only able to read the content. The generic user interface uses Twitter Bootstrap[1] that contains HTML and CSS design templates for forms, buttons, navigation and other interface components. AngularJS[2] is used to declare dynamic views by extending HTML vocabulary with new directives. Main advantage of AngularJS is its data binding that automatically updates the view whenever the underlying models changes. Another advantage that is very valuable for the development of our software solution is the reusability of directives in AngularJS that can be executed in different places. We incorporated AngularStrap[3] in order to ease the integration between Bootstrap and AngularJS. It provides a set of native directives that can be directly reused in the interface, e.g., the worklist feature uses the directive for the side window provided by AngularStrap.

The mobile interface is a completely separate front-end implementation compared to the generic interface. This is necessary since it has a limited set of features compared to the generic interface that is used for the desktop version. Depending on the device that is making the request in the browser, the main controller switches to the right implementation. The mobile interface is implemented using Angular Material[4], which is described in detail in Subsection 4.3.1. Main reason for this decision is that Angular Material is particularly suitable for interactions with mobile devices. Finally, tailors work with a subset of the generic user interface that is only visible for them and it could also be entirely extracted as separate implementation. In this interface the CMMN workbench that allows the definition of work templates is implemented. The workbench is described more detailed in Subsection 4.2. This subset of the user interface requires a library for visual diagrams to support the modeling using the CMMN standard. At the time when the software solution was developed there were no existing implementation of the standard available. Therefore, we developed an own CMMN editor based on the JointJS[5] diagramming library. It basically allows to create interactive generic diagramming tools for different purposes and it is written entirely in JavaScript.

[1]`http://getbootstrap.com/`, last accessed on: 2015-06-06
[2]`https://angularjs.org/`, last accessed on: 2015-06-06
[3]`http://mgcrea.github.io/angular-strap/`, last accessed on: 2015-06-06
[4]`https://material.angularjs.org/`, last accessed on: 2015-06-06
[5]`http://www.jointjs.com/`, last accessed on: 2015-06-06

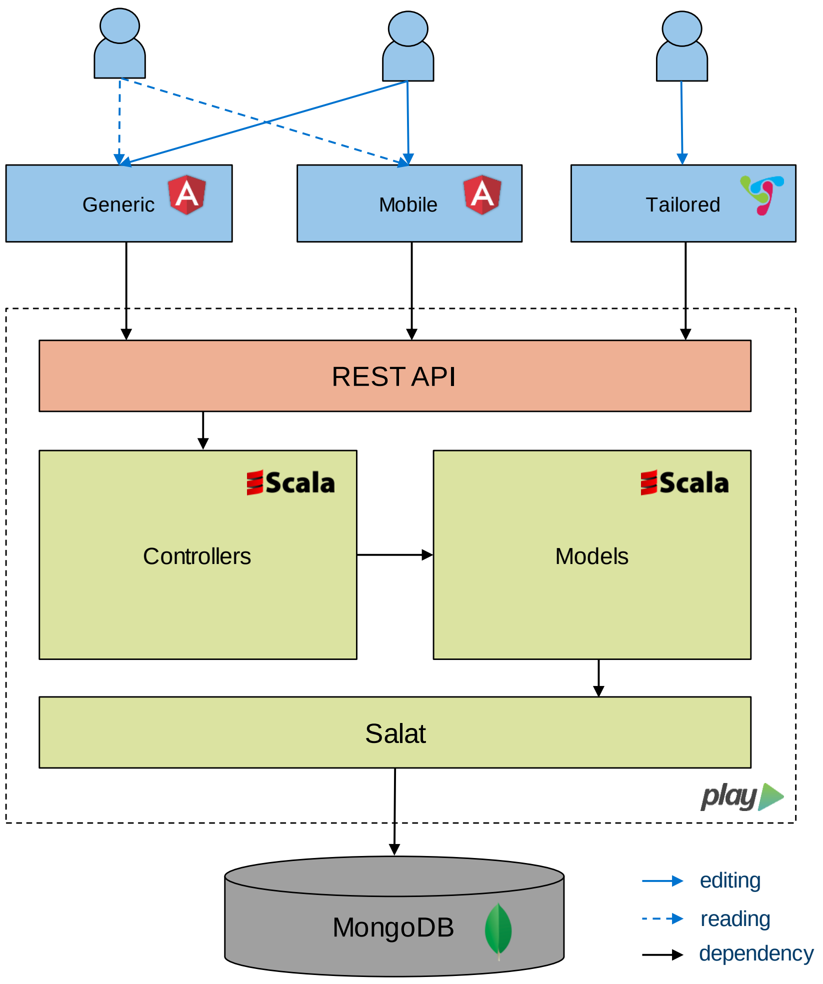


Figure 4.1.: Architecture of the implemented software solution

In this thesis we provide the implementation for these basic three user interfaces which are independent of a specific application domain. Due to our extensible architecture new interfaces can be easily build for domain-specific application scenarios, e.g., for healthcare, project management, or software engineering. In particular the interface for end-users could be developed entirely new or adapted for different domains. Although we are convinced that the generic user interface can be used for most of the domain specific application scenarios as well, it might be necessary to provide another design or taxonomy that is easier to understand for end-users. In contrast to the generic and mobile interface that might be specific for end-users, the tailored interface will likely be the same across all application scenarios since it is based on the CMMN standard. The separation between the frond- and back-end is loosely coupled to make the development of new interfaces comfortable. Our server exposes various interfaces that are based on the REST architectural style presented by Fielding in his PhD thesis [Fi00]. All interfaces in the API of our server communicate via JSON[6] and are described more detailed in Subsection 4.1.3.

[6]`http://www.w3schools.com/json/`, last accessed on: 2015-06-06

Our server is implemented in the play framework[7] that is described in detail in Section 4.1.2. Main advantages of this framework are its scalable stateless architecture and the comprehensive ecosystem that provides plenty of modules. It also specifies a basic software architecture that already divides all classes into controllers and models. We decided to implemented the system in Scala, which is an object-oriented and functional programming language that is explained in [Oa04]. The models are only used to define the data structure including required functions for the basic create, update and delete operations. The entire logic is implemented strictly in controllers and we were anxious not to mix up both layers we each other. The dependency is one directional from controllers to models, i.e., calls from the model layer upwards are forbidden. All models are stored in the database using the Salat[8] library. The database for the software solution is the document-oriented MongoDB[9]. Salat is a bidirectional serializer for case classes that are implemented in the Scala programming language. For this purpose it builds directly on cashbash[10], which is the internal query language from MongoDB. Using MongoDB provides several advantages for the software solution developed in this thesis. First of all it supports dynamic schemas much better than relational structures provided by traditional databases. Another reason is its ability for horizontal scaling based on sharding that allows the usage of distributed parallel instances. Files are not stored directly in the database of the prototypical implementation and always reside in the file system of the operating system within a static folder.

4.1.2. Introduction to the Play Framework

According to the philosophy of the play framework[11] it follows five design principles. First, it is build for asynchronous programming so that long-living requests that are based on an event model are possible. Actors are implemented as core concept in the framework that allow the simple development of concurrent entities. These entities handle messages asynchronously that are used with a pattern matching to define the behavior of actors. Second, it is focused on type safety through its statically typed programming language in the back- and front-end. Through this design decision, the compiler is able to detect errors in the template language as well. Third, it supports Java and Scala as built-in programming languages. Scala is in the core of the framework and not only available as separate module. Fourth, the powerful build system is easy to use with the three simple commands new, run and start that are sufficient for the deployment. Creating standalone versions of a play web application can be done with the dist command. With this command the framework packages all required files and generates an execution script that can be started without any additional configuration. The build system allows hot code reloading and displays error messages in the browser to improve efficiency of developers. In case more sophisticated build and deployment is necessary, the underlying interactive build tool sbt can be adapted. Finally, the play framework is not restricted to SQL databases since any store driver can be used within the framework, e.g., the salat driver for MongoDB that can be used with the Scala programming language.

[7] `https://www.playframework.com/`, last accessed on: 2015-06-06
[8] `https://github.com/novus/salat/`, last accessed on: 2015-06-06
[9] `https://www.mongodb.org/`, last accessed on: 201
[10] `https://github.com/mongodb/casbah`, last accessed on: 2015-06-06
[11] `https://www.playframework.com/documentation/2.4.x/Philosophy`, last accessed on: 2015-06-06

4.1.3. Application Programming Interface

The API exposes various interfaces that are used by the different front-ends as described in Figure 4.1. In future work these interfaces could be also used to integrate other external systems with our software solution, e.g., for the integration of additional data sources. Figure 4.2 illustrates how incoming http requests to the server are mapped to controllers according to the routes file that is defined in the play framework, i.e., in this routes file every URL is mapped to exactly one controller. Given that the API exposes all necessary functions and is well documented, it should not be necessary for developers to understand the internals of the server implementation to create new frond-ends. Therefore, we provide a complete description of the most important API methods for the software solution developed in this thesis in Appendix A. For the sake of brevity only core methods are described in this appendix. Most of the controllers that are introduced provide some additional requests that are not explained in this thesis and only briefly mentioned.

In general the available controllers can be grouped in four categories. Two categories are related with controllers for work plans and work templates. The third category is related with user management and the fourth category with the social feed feature. Every API method is described with the used http method which can be GET, PUT, POST and DELETE in our case. We provide the controller that is invoked by the API method and explain the expected response or impact of the controller method in the server. Responses of controllers are always transmitted in the JSON format. Controllers implement the behavior and interact with models that are responsible for the efficient storage of entities in the database. The interaction of controllers with the data model layer is described more detailed in the subsequent Section 4.1.4. Finally, the internal implementation of the controller with its behavior is briefly explained.

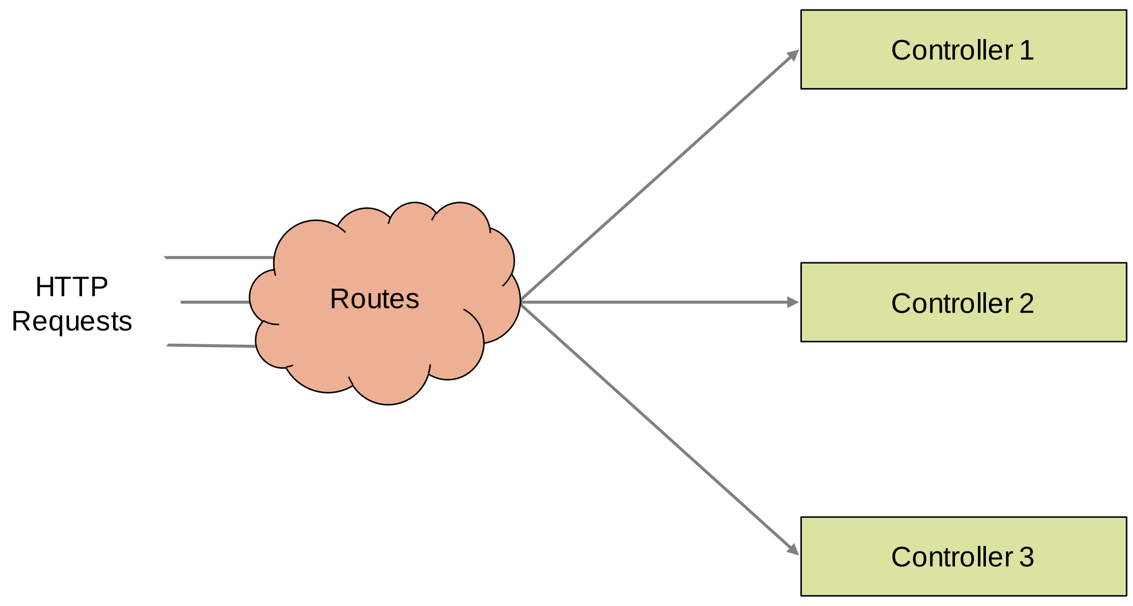


Figure 4.2.: Routing of external requests to controllers in the play framework server

4.1.4. Data Model

The controllers that are invoked by the API execute their business logic based on the models in the play framework server. We carefully avoided calls upwards from models to controllers to improve the maintainability of the server implementation, i.e., to allow the replacement of controllers or models in the future. In this section, we introduce our data model that is implemented in these models. For a better understanding we will start explaining the high-level package structure of the models first. After this introduction these packages are explained in detail with the attributes and methods of all classes. Figure 4.3 shows an overview of the implemented data model with the main packages. Due to the complexity of the data model attributes, methods and some relationships between the packages are hidden. Every package is annotated with the name and section number in which it is explained in detail.

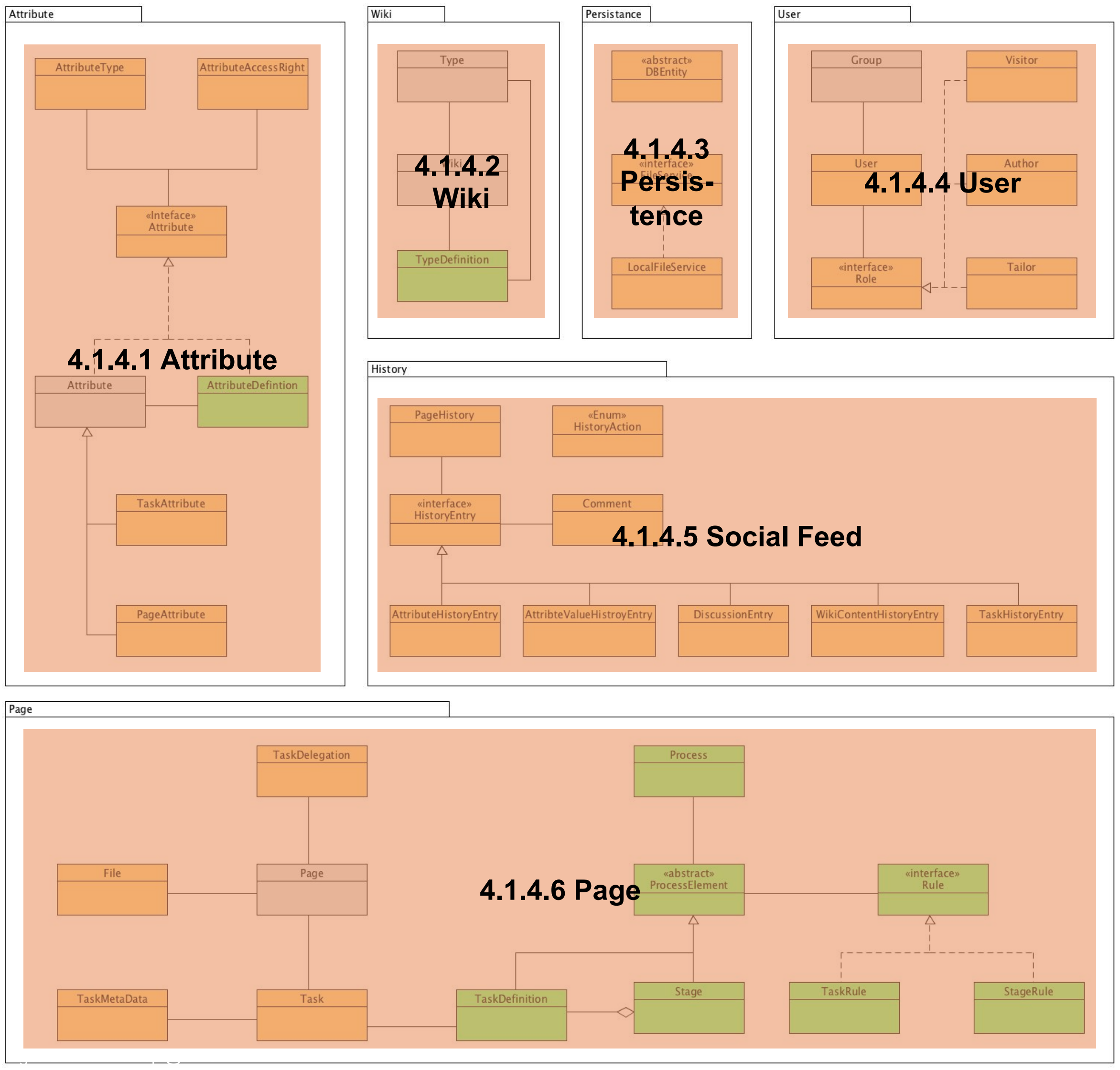


Figure 4.3.: Implemented data model with the packages that are mapped to sections

Everything related to the representation of attributes in the data model is located in the package *attribute*. This package contains an interface that is associated with an *AttributeType* and an *AttributeAccessRight*. The implementation of the attribute interface can be an *Attribute* or *AttributeDefinition*, wheres attributes can have a reference to exactly one AttributeDefintion to describe the relationship to work templates. Although the basic concepts for Attributes and AttributeDefinitions were already proposed by Hybrid Wikis in [Ne12], we extend them with additional classes necessary to represent attribute types, access rights and task attributes. *TaskAttributes* are necessary to manage attributes that are assigned to tasks. The implementation of the *Wiki* package is very close to the conceptual model introduced in Section 3.2. It only consists of a *Type*, *Wiki* and *TypeDefinition* that are linked with each other. The attribute package has a dependency to the wiki package with a reference from AttributeDefinitions to TypeDefinitions. The *persistance* package contains several base classes that provide functionality that is required in all other class of the data model, e.g., the serialization to and from JSON for the controllers. In addition, the persistence package provides a *FileService* that is used to store and retrieve files that are added as values for attributes.

The *user* package contains several classes to represent *Users* and *Groups*. In addition, every user also has roles for the various TypeDefinitions assigned, e.g., visitor, author, or tailor. These roles are not distinct and a user can be part of several roles at the same time for a TypeDefinition. The roles are bound to TypeDefinitions because a user might be an author for one TypeDefinition and tailor or visitor for another TypeDefinition. In the current implementation roles are only bound to users and not to groups, whereas we consider this as a valuable extension in the future, i.e., all users that belong to a certain group are authors for a TypeDefinition. All required classes for the social feed feature are summarized in a package called *history*. Every page has it own *PageHistory* that contains its entire history starting with the creation of the page. The package contains an interface *HistoryEntry* that is implemented by a variety of different types of history events. Examples for history events are attributes, attribute values, wiki pages and tasks. Discussions in the feed are also treated as HistoryEntry since they have the same functionality. The only difference to the other HistoryEntry classes is that they are manually created by users. The basic set of history events that is currently implemented can be easily extended in case new types of events are necessary in the future. In addition to the history entries this package also provides a class for *comments*. This class is bound to the *HistoryEntry* interface since every entry in the social feed can be commented.

Finally, the *page* package contains all classes that are related to the representation of wiki pages. Similar to the packages for the attribute and wiki, this package consists of classes that are necessary for the schema and instance level in the current stage of the implementation. In future work all classes that are related to the schema level could be extracted to separate packages. This new package might contain all classes that are necessary for the schema from the current packages page, attribute and wiki. The class *Page* is used to store wiki pages with a relationship to *Files*, *Tasks* and a *Type*. The class File is used to store files that are not structured with attributes on the wiki page, e.g., files that are used as attachments or shown in the text of the wiki page. Every Page has a class called *Process* assigned that is determined by its TypeDefinition and contains the entire process structure consisting of *Rules*, *Stages* and *TaskDefinitions*. The package contains a class *TaskMetaData* for every Task that is assigned to a Page. The class *TaskDelegation* is used to store information on users that delegate tasks.

4.1.4.1. Attribute

In the following all attributes and methods of the classes in the package Attribute are explained in detail. Figure 4.4 shows the detailed data model of the Attribute implementation. The Attribute interface has an attribute for the *name* and two methods for the *equals* and *hashCode* operation. The equals method compares the name and type of two attributes to determine whether the attributes are equal. Every Attribute has exactly one *AttributeType* and *AttributeAccessRight* assigned. A simple string is used to store the attribute type in the class AttributeType, e.g., date, enum, string, or type. In case another wiki page is referenced with this attribute the type of this wiki page is stored.

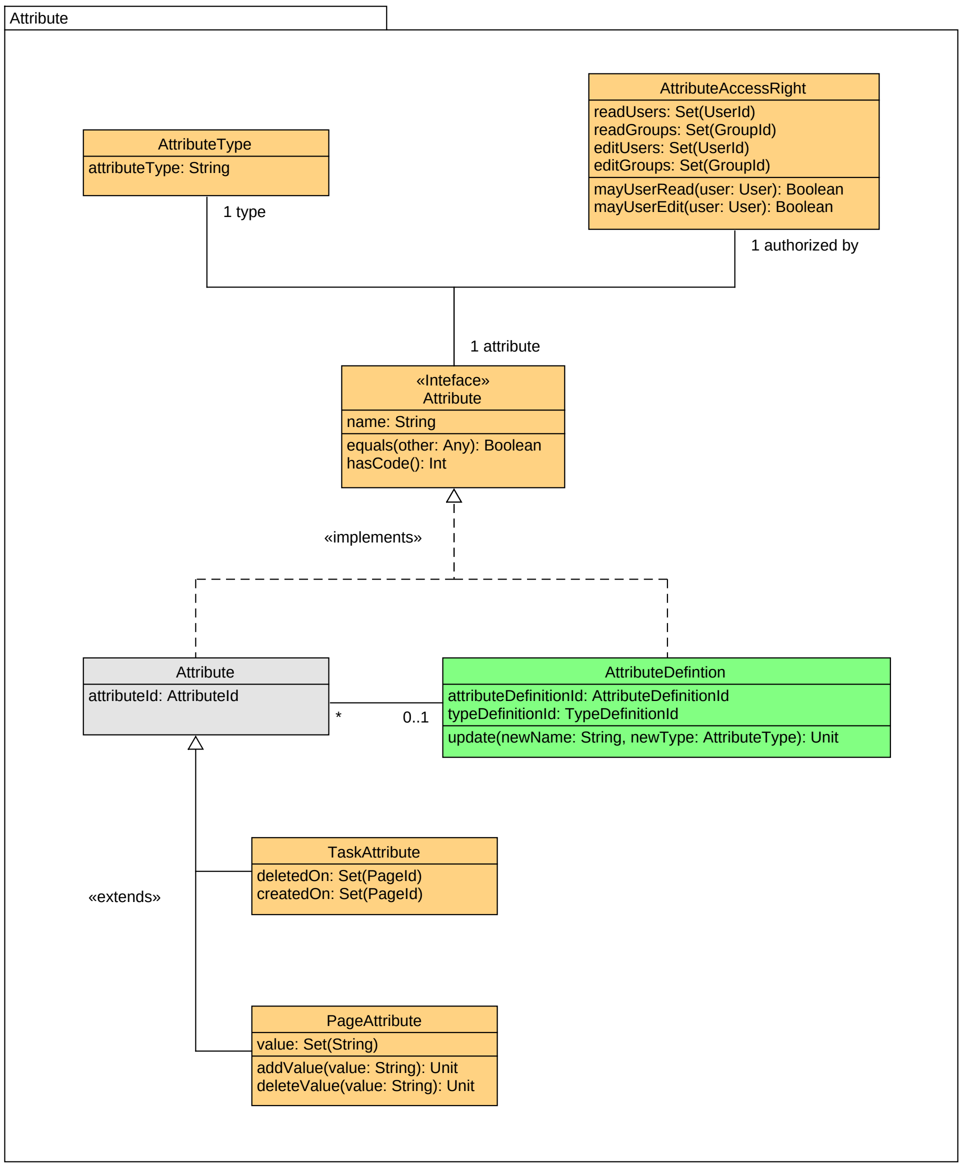


Figure 4.4.: Implementation of the classes for the attribute concept

The class AttributeAccessRight consists of four sets that contain the ids of the groups and users that have access. Users and groups can have read and edit access to an attribute, e.g., in case a user is authorized to edit the attribute his userId is stored in the set *editUsers*. Every time users request read or write access to an attribute two method from the class AttributeAccessRight are invoked. The first method *mayUserRead* has the requesting user has parameter and returns a boolean as response. This return value determines whether the user is allowed to access the attribute. Similarly, the method *mayUserEdit* determines if the user is allowed to edit the attribute. Both classes *Attribute* and *AttributeDefinition* implement this attribute. An Attribute might be assigned to exactly one AttributeDefinition to support the late data modeling requirement. In the same way the AttributeDefinition has a reference to a TypeDefinition with the typeDefinitionId. Users that are authorized to edit the AttributeDefinition can perform an *update* method to change the name or type. Attributes are extended to *TaskAttributes* and *PageAttribute*. The PageAttribute is stored on wiki pages with a set of values that are stored as string. This class provides two methods that are used to add or delete a value from the PageAttribute. The TaskAttribute is created every time a PageAttribute is assigned as mandatory attribute to a task. This class contains two sets that store pageIds of wiki pages in which this attribute was created or deleted for a task.

4.1.4.2. Wiki

The package wiki consists of three classes that are illustrated with their implemented data model in Figure 4.5. The *Wiki* has a name that is shown in the explorer for the navigation. The data structure of the wiki stores the date when it was *created* and the *creatorId* of the user that initially created the Wiki. The date of the last edit operation on the Wiki is stored in the *lastEdit* attribute. Every Wiki has an attribute *text* that stores the content of the Wiki. Three methods are provided by the Wiki class that can be used to access to content of the Wiki. The method *ChildrenPages* returns an iterator that can be used to iterate through direct children pages, i.e., pages that have no parent page. The method *descendantPages* returns an iterator with all pages within the wiki and not only the direct children. The method *childrenTypes* returns all types that belong to this wiki with an iterator. These types are implemented with the class *Type* that have a unique *typeId*. This typeId is stored by wiki pages that have this Type assigned. The class Type provides several methods to maintain the attributes that are assigned to the Type. The method *getAttribute* provides all PageAttributes that are assigned on wiki pages with this type. Several different methods allow the adding and deleting of attributes on wiki pages for this type. For this methods the pageId needs to be provided since the methods only affect the page with this pageId and not all wiki pages that have this Type assigned. The different implementations are mainly provided for convenience reasons, since the only distinction between them is the set of parameters that is required, i.e., the methods can be executed with the id of the Attribute or the name and type. Similarly, the class *TypeDefinition* provides several similar attributes and methods that are used to describe the schema level. Every Type can reference at most one TypeDefinition within a Wiki that is identified with its unique *typeDefinitionId*. In contrast to the Type, the AttributeDefinitions that are added to the TypeDefinition are shown on all wiki pages that have this TypeDefinition assigned. The Type and TypeDefinition classes are used to maintain the data structure, while tasks with their dependencies are maintained in a separate Process class that is described with the page in Section 4.1.4.6.

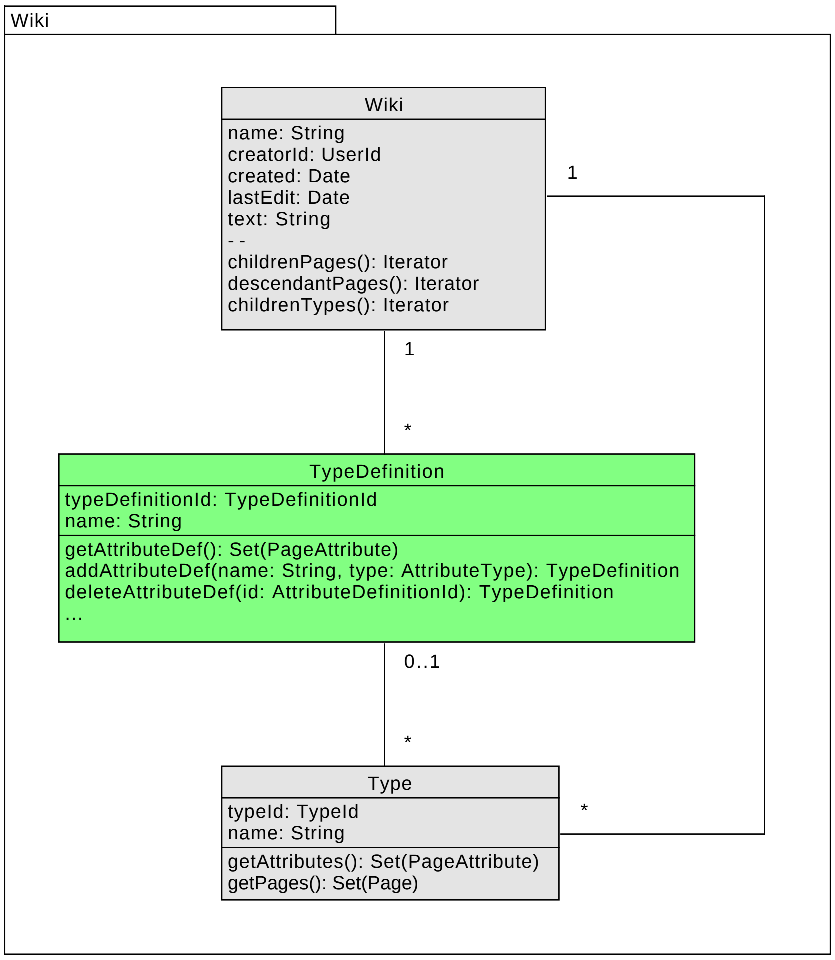

Wiki
Wiki
name: String
creatorId: UserId
created: Date
lastEdit: Date
text: String
- -
childrenPages(): Iterator
descendantPages(): Iterator
childrenTypes(): Iterator
1
1
*
TypeDefinition
typeDefinitionId: TypeDefinitionId
name: String
getAttributeDef(): Set(PageAttribute)
addAttributeDef(name: String, type: AttributeType): TypeDefinition
deleteAttributeDef(id: AttributeDefinitionId): TypeDefinition
...
0..1
*
Type
typeId: TypeId
name: String
getAttributes(): Set(PageAttribute)
getPages(): Set(Page)
*

Figure 4.5.: Implementation of the concepts in the wiki package

4.1.4.3. Persistence

All models that are stored in the database extend at least one class from the package *Persistance* that is shown in Figure 4.6. The abstract class *DBEntity* is extended by all other models presented in this section. This class provides some basic methods and logic that is necessary for the interaction with the MongoDB database, e.g., the attributes *dao* and *collection*. The methods of this class are important to retrieve and store data of the models from the database. Two methods have to be overwritten by all models to *read* and *write* DBEntity classes that are translated to JSON and vice versa. The method *findByIds* also needs to be implemented by all models since it gathers DBEntity by their id, e.g., instances with the specified ids of the class Type. The method *beforeRemove* has to be implemented by some models that require a deletion process that can consists of more than one step, e.g., deleting a wiki page requires the deletion of subpages, attributes and tasks that are attached before the wiki page can be removed from the database. Several remove methods are provided to delete models from the database based on the provided id or directly as member method of the model class.

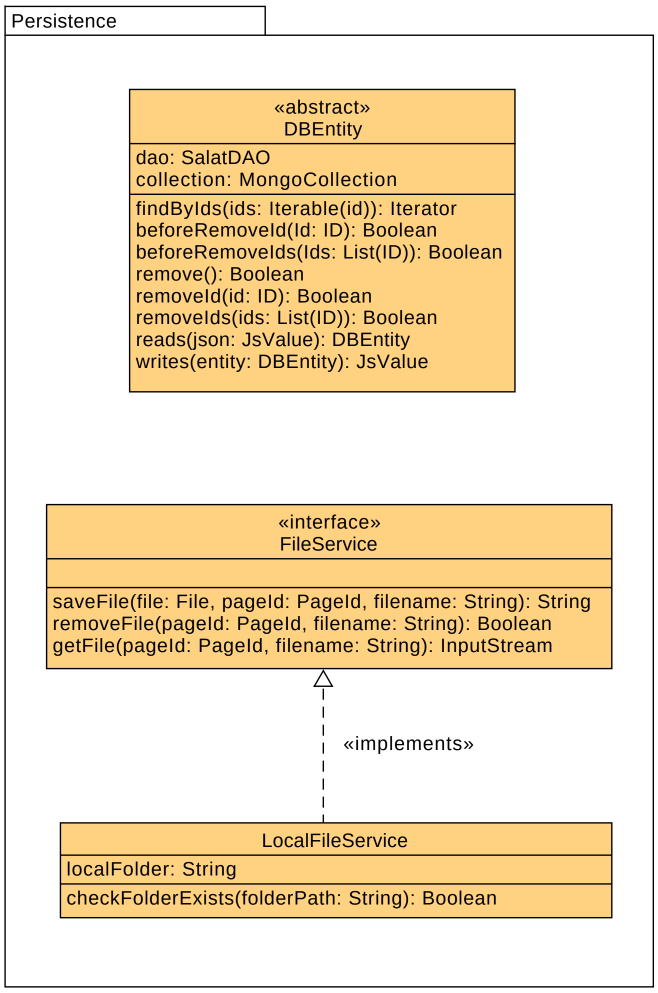

Persistence
«abstract»
DBEntity
dao: SalatDAO
collection: MongoCollection
findByIds(ids: Iterable(id)): Iterator
beforeRemoveId(Id: ID): Boolean
beforeRemoveIds(Ids: List(ID)): Boolean
remove(): Boolean
removeId(id: ID): Boolean
removeIds(ids: List(ID)): Boolean
reads(json: JsValue): DBEntity
writes(entity: DBEntity): JsValue
«interface»
FileService
saveFile(file: File, pageId: PageId, filename: String): String
removeFile(pageId: PageId, filename: String): Boolean
getFile(pageId: PageId, filename: String): InputStream
«implements»
LocalFileService
localFolder: String
checkFolderExists(folderPath: String): Boolean

Figure 4.6.: Implementation of the persistence package

Attribute values and wiki pages can have files attached that can be used to store arbitrary unstructured content. In case the degree of structure is very low, wiki pages might have only files attached without any structured information that is stored in attributes similar to a simple content management system. On the other hand files are also important to document results for tasks that are developed with external tool solutions. To support the handling of files the persistence package provides a basic interface called *FileService*. This interface provides three generic methods to *save files*, *remove files* and *get files*. In the current implementation we only implemented one class based on this interface. The class *LocalFileService* is used by the software solution to store files on the local filesystem. The attribute *localFolder* provides the path the folder that is used to store the files. With the method *checkFoldersExists* the path specified as parameter is validated. In future work additional implementations for the FileService could be provided to tie distributed file systems in the software solution. Another possible extension for the future is the implementation of version management in the FileService.

4.1.4.4. User

The implementation of the classes necessary for the user management is illustrated in Figure 4.7. Similar to Hybrid Wikis presented by Neubert in [Ne12] our software solution provides the notion of groups. The class group has an attribute *delegatedTasks* to store tasks that are delegated by other users to this group. The concept of Hybrid Wikis is extended with a more sophisticated class to represent *Users*. The basic user information stored in this class are the *name*, *email* and *password* that are necessary to authorize users in the system. Additionally, users can mark wikis and wiki pages as favorite to highlight them in the user interface. Users can be added and removed from groups with two methods that are implemented in the User class. In the current implementation the skills and the assigned tasks are not stored in the data structure of the user. This information is queried by the user controller with every method invocation based on the tasks that are assigned to the user. In future work information about skills could be stored in the User class to avoid unnecessary computations by the server. Every user can have several roles for the Types in the system assigned, e.g., users can have more than one role that can be different for the Types. Three different classes for roles are implemented which are called *Visitor*, *Author* and *Tailor*.

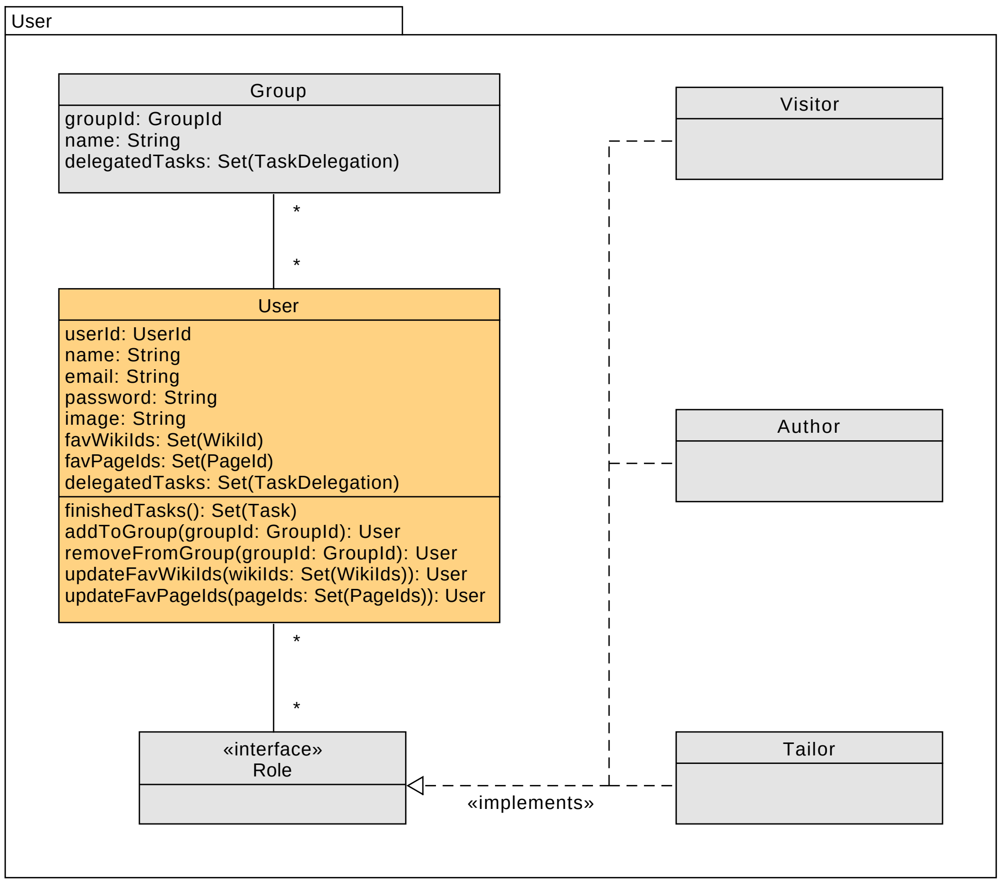

User
Group
groupId: GroupId
name: String
delegatedTasks: Set(TaskDelegation)
*
*
User
userId: UserId
name: String
email: String
password: String
image: String
favWikiIds: Set(WikiId)
favPageIds: Set(PageId)
delegatedTasks: Set(TaskDelegation)
finishedTasks(): Set(Task)
addToGroup(groupId: GroupId): User
removeFromGroup(groupId: GroupId): User
updateFavWikiIds(wikiIds: Set(WikiIds)): User
updateFavPageIds(pageIds: Set(PageIds)): User
*
*
«interface»
Role
«implements»
Visitor
Author
Tailor

Figure 4.7.: Implementation of the classes for user management

4.1.4.5. History

The data model for the social feed is implemented with the history package and illustrated with all classes in Figure 4.8. Since wiki pages are the core entities in our software solution, every page has its own history that is stored in the class *PageHistory*. Every PageHistory has its own unique *pageHistoryId* and *pageId* that references the wiki page that is assigned to it. All history entries are stored in an attribute as list and the PageHistory class provides two methods to add new entries. All history entries have to extend the base class *HistoryEntry* that provides various methods and attributes that are common for all entries. Every entry has a *time* attribute that stores the current date when the action that triggered the history entry occurred. All history entries are created manually through a user that performs an action in the system, e.g., finishing a task or uploading a new document in the system. For this purpose the *userId* of the user is stored for every history entry. All history entries can be rated by other users with the attribute *likes*. New history entries in the social feed can be highlighted for users or a small badge with the number of new tasks is shown in the user interface. For this purpose a set of userIds is stored with the attribute *seenUsers* since this information needs to be captured for every user. In case no user has seen this history entry the attribute *newEntry* is set to true.

The attribute *historyAction* stores one value from the enum *HistoryAction*. It consists of the name of the basic actions that are possible for users, e.g., adding, delegating, or skipping tasks. All history entries can be commented by users in the system, whereas the comments are stored in a separate class called *Comment*. This class stores the *creatorId* of the user that initially added the comment and the *text*. Every comment has a unique *commentId* that is used by the HistoryEntry to reference the comments that are assigned. Two dates are stored for every comment called *created* and *lastEdit*. The creator of a comment is able to invoke the method *updateText* to make changes on his comment. Every call of this method updates the attributes lastEdit. In the current implementation five classes are implemented that extend the HistoryEntry base class, whereas new child classes can be created in the future in case the social feed needs to be extended. The child class *AttributeHistoryEntry* is instantiated every time a new attribute is added or changed on a page by authors. It contains an additional attribute *attributeName* that is shown with the HistoryAction in the feed, e.g., user «name of the user» removed attribute «name of the attribute».

The class *AttributeValueHistoryEntry* is used to document actions that are related to changes of attribute values, e.g., adding or deleting values from attributes. In case an attribute value is changed many times consecutively only one history entry is generated. In addition to the name this history entry stores the value of the attribute. Discussion entries are stored in the class *DiscussionEntry* with an attribute that stores the *content* as string. The discussion can be used to make announcements or bring up a question for the community. While the other history entries are indirectly created by other actions that are performed in the system, the discussion entry is directly created in the social feed. The class *WikiContentHistoryEntry* stores changes that are performed on the content of wikis and wiki pages. The *TaskHistoryEntry* stores actions that are related to tasks. A history event is created for tasks that are delegated to other users. The name of the user is stored in the attribute *delegatedTo*. Tasks with an updated progress also generate a history event and in these cases the new progress is stored in the attribute *updatedTo*.

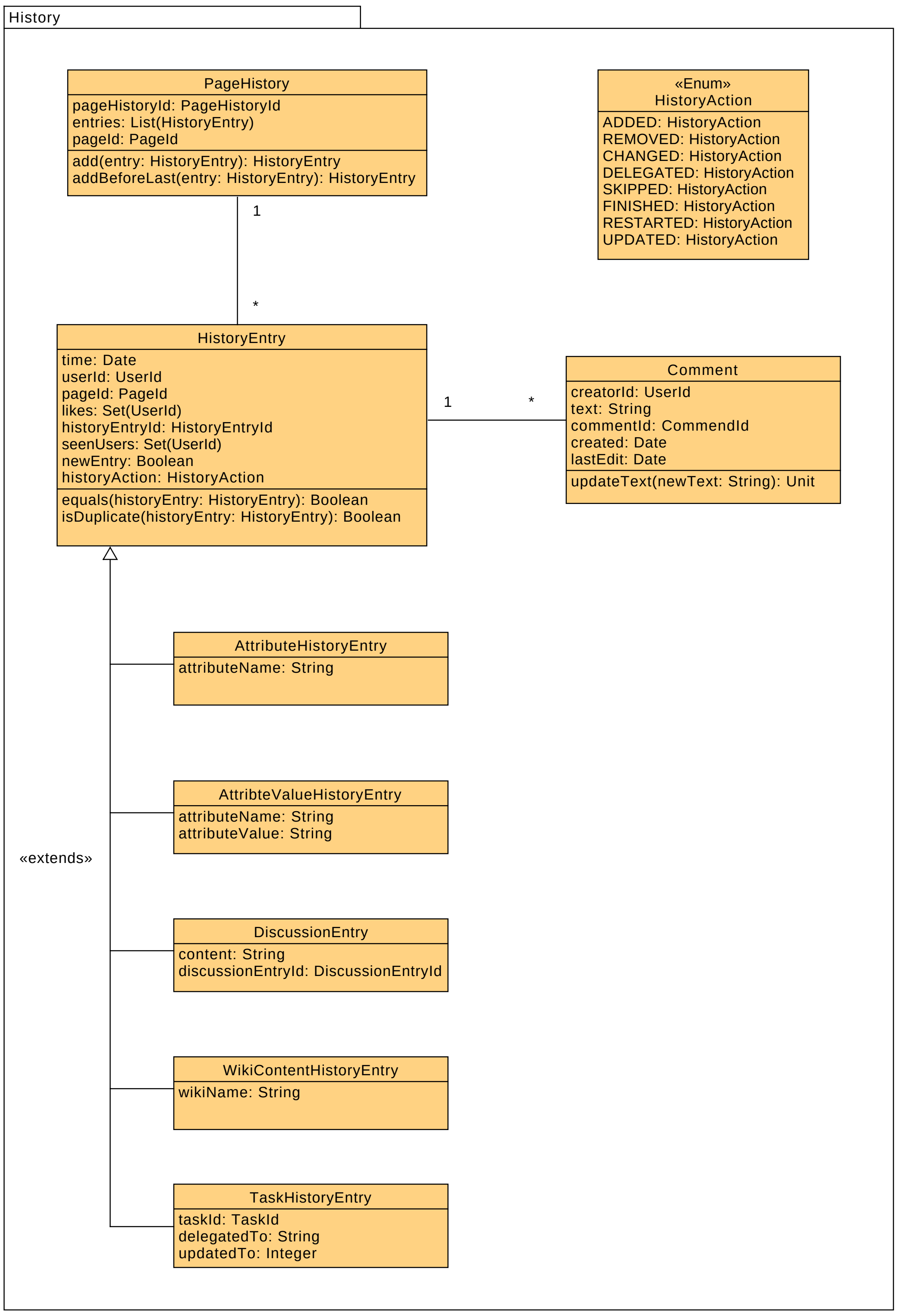


Figure 4.8.: Implementation of the history package that is used for the social feed

4.1.4.6. Page

The data model for the implementation of the wiki page is illustrated in Figure 4.9. Every page has several *Files* and *TaskDelegations* associated. Files are stored with the FileService from the persistence package. The TaskDelegation is created for every task that has been delegated on this wiki page. Every Page stores history information in several attributes that can be shown in the user interface, e.g., *creatorId*, *lastEdit* and *lastEditorId*. The content of the wiki page is stored as string in the attribute *text*. In *parentIds* a list of pageIds for all parent pages is stored. In the current implementation the attribute *currentTaskId* stores only one task that is enabled, whereas this can easily be extended to store the currentTaskId for every author of the wiki page. The attribute *progress* contains the average progress of all tasks assigned to this wiki page to avoid the computation with every request of the wiki page. All child pages of a wiki page can be returned with the method *children*. Updating the content of the page is possible using the method *updateText*. Similarly, several methods are provided for the class Page, e.g., to update the progress or add attribute to the wiki page. For the sake of brevity not all methods implemented in the Page class are described in this thesis.

Pages can have several *Tasks* assigned and every task has exactly one class *TaskMetaData*. The class TaskMetaData contains the start and end dates, progress and expertises for the associated Task. The attributes *overdueSubtasks* and *inconsistentSubtasks* contain sets with TaskIds. In case the associated task is already completed, the attributes *finishedAt* and *finishedByUser* are populated. The class Task contains a string for the *name* of the task and the unique *taskId*. Mandatory attributes that are assigned to the task are stored in *attributes* with a set of TaskAttributes. The position of the task in the CMMN workbench is stored in the two attributes *posX* and *posY*. The access rights for the task are stored in several attributes that contain a set of userIds or groupIds, e.g., *SkipUsers* contains the userIds of the users that are allowed to skip this task. Three methods are implemented in the class to evaluate whether a users has certain access rights for a task, e.g., the method *mayUserExecute* checks if the user with the userId is stored in the attribute executeUsers or if the user is member of a group that is stored in executeGroups. The methods *isOverdueOnPage* and *isInconsistentOnPage* evaluate the state of the task based on the associated TaskMetaData.

Tasks can be associated to one *TaskDefinition* at most, e.g., in case the task is created through the instantiation of a work template. The class *Stage* can contain several TaskDefinitions to structure the work template. Stages and TaskDefinitions extend an abstract class *ProcessElement*. These ProcesseElements can be linked through rules to define logical dependencies, e.g., TaskDefinitions within a Stage are enabled after a preceding TaskDefinition is finished. The ProcessElements with their Rules are associated to the class *Process*. Every TypeDefinition has exactly one Process associated that describes the entire process structure. In our approach only tailors are able to make changes on instances of the class Process based on tasks and attributes that are provided by authors. For this purpose the class Process provides methods to access tasks created by authors on certain wiki pages, e.g., *getAvailableTasks* that returns all Tasks for the wiki page with the id pageId. The method *getFinishedTasks* returns a list of completed Tasks for the wiki page with the provided pageId. The creation of new TaskDefinitions, Stages and Rules is performed in the interactive CMMN workbench that is explained Section 4.2. For this workbench several methods are provided by the Process class, e.g., *newStage*, *newTaskDefinition* and *newRule*.

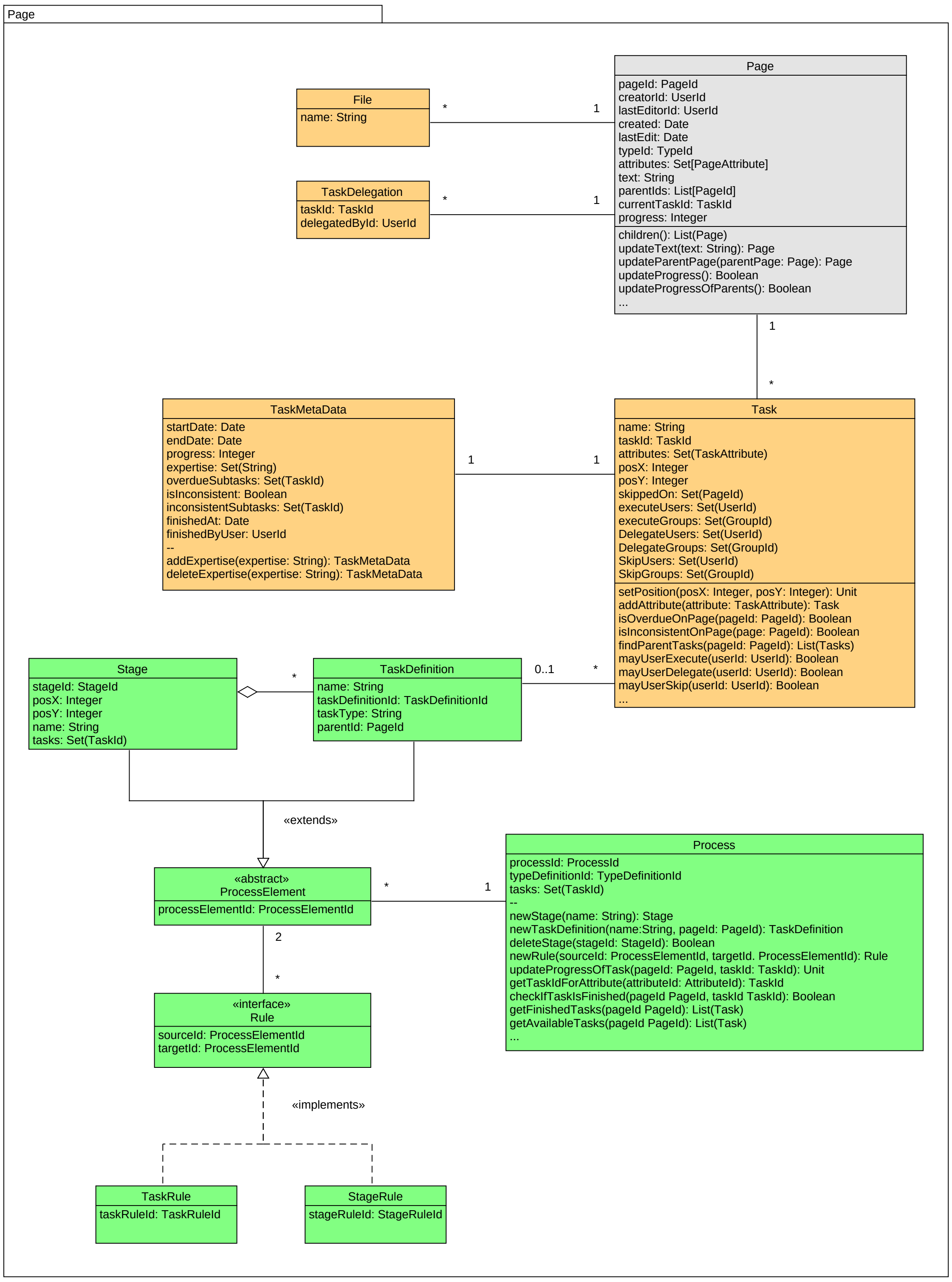

Page
File
name: String
TaskDelegation
taskId: TaskId
delegatedById: UserId
Page
pageId: PageId
creatorId: UserId
lastEditorId: UserId
created: Date
lastEdit: Date
typeId: TypeId
attributes: Set[PageAttribute]
text: String
parentIds: List[PageId]
currentTaskId: TaskId
progress: Integer
children(): List(Page)
updateText(text: String): Page
updateParentPage(parentPage: Page): Page
updateProgress(): Boolean
updateProgressOfParents(): Boolean
...
*
1
*
1
1
*
TaskMetaData
startDate: Date
endDate: Date
progress: Integer
expertise: Set(String)
overdueSubtasks: Set(TaskId)
isInconsistent: Boolean
inconsistentSubtasks: Set(TaskId)
finishedAt: Date
finishedByUser: UserId
--
addExpertise(expertise: String): TaskMetaData
deleteExpertise(expertise: String): TaskMetaData
1
1
Task
name: String
taskId: TaskId
attributes: Set(TaskAttribute)
posX: Integer
posY: Integer
skippedOn: Set(PageId)
executeUsers: Set(UserId)
executeGroups: Set(GroupId)
DelegateUsers: Set(UserId)
DelegateGroups: Set(GroupId)
SkipUsers: Set(UserId)
SkipGroups: Set(GroupId)
setPosition(posX: Integer, posY: Integer): Unit
addAttribute(attribute: TaskAttribute): Task
isOverdueOnPage(pageId: PageId): Boolean
isInconsistentOnPage(page: PageId): Boolean
findParentTasks(pageId: PageId): List(Tasks)
mayUserExecute(userId: UserId): Boolean
mayUserDelegate(userId: UserId): Boolean
mayUserSkip(userId: UserId): Boolean
...
Stage
stageId: StageId
posX: Integer
posY: Integer
name: String
tasks: Set(TaskId)
*
TaskDefinition
name: String
taskDefinitionId: TaskDefinitionId
taskType: String
parentId: PageId
0..1
*
«extends»
«abstract»
ProcessElement
processElementId: ProcessElementId
*
1
Process
processId: ProcessId
typeDefinitionId: TypeDefinitionId
tasks: Set(TaskId)
--
newStage(name: String): Stage
newTaskDefinition(name:String, pageId: PageId): TaskDefinition
deleteStage(stageId: StageId): Boolean
newRule(sourceId: ProcessElementId, targetId. ProcessElementId): Rule
updateProgressOfTask(pageId: PageId, taskId: TaskId): Unit
getTaskIdForAttribute(attributeId: AttributeId): TaskId
checkIfTaskIsFinished(pageId PageId, taskId TaskId): Boolean
getFinishedTasks(pageId PageId): List(Task)
getAvailableTasks(pageId PageId): List(Task)
...
2
*
«interface»
Rule
sourceId: ProcessElementId
targetId: ProcessElementId
«implements»
TaskRule
taskRuleId: TaskRuleId
StageRule
stageRuleId: StageRuleId

Figure 4.9.: Data model of the implemented classes in the page package

4.1.5. Assignment of Attributes to Tasks

Attributes can be assigned to tasks as mandatory deliverables that have to be created, i.e., tasks are automatically completed after all mandatory attributes are entered by the user. Attributes are assigned to tasks with a drag and drop mechanism that requires no programming capabilities. Authors can easily drag an attribute from the wiki page and drop it on the task that has to create the value for this attribute. Figure 4.10 shows a sequence diagram with the communication between the involved objects. The entire process starts with a user that clicks on an attribute of the wiki page. This triggers the method *openAside* in the view controller that opens the side window with all tasks that are assigned to the current wiki page and visible for the user. After the user dropped the attribute on the desired task, the method *addTaskAttribute* is started in the page controller. Both objects are abbreviated with viewCtrl and pageCtrl. These viewCtrl and pageCtrl are implemented in the frond-end with AngularJS. The pageCtrl in the frond-end formulates a http POST request with the name and type of the attribute as well as the taskId of the task. This request is mapped to the *PageController* in the server based on the mapping defined in the routes file.

In the PageController the method *addTaskAttribute* is invoked with the parameters that are provided from the pageCtrl in the front-end. This method handles the entire logic that is necessary for the assignment of the attribute to the task. In the first step the wiki page is gathered with the *findOneById* method. The required pageId for this method is provided in the URL of the http request. In the second step the task that is selected by the user is gathered in a similar way. The taskId for the gathering of the task is stored in the body of the request. After the task is gathered the page controller invokes the *addAttribute* method with two parameters. The parameters are the name of the attribute and the type of the attribute, which are provided in the body of the request. The page controller stores the task by invoking the save method as soon as the attribute is added. After the assignment of the attribute is stored in the task, the page controller updates the progress. This is necessary since the attribute assignment might have an influence on the progress, e.g., in case the assigned attribute already has a value or the task already has a progress that is larger than 0%. In the first case the attribute can have another wiki page with tasks assigned. The progress of these subtasks needs to be incorporated for the new task.

Finally, the page controller updates the page progress by invoking the method *updateProgress* of the page instance that was gathered previously. This method computes the page progress based on the tasks and subtasks that are assigned to the page and stores the result in an attribute of the page. After the addTaskAttribute method is completed it returns the http status code ok. The pageCtrl waits for this response with a success method that triggers the printing of a log message in the console and returns. After the pageCtrl is completed the viewCtrl invokes the method *loadPageData*. The loadPageData method reloads the entire wiki page by requesting the data structure of the page controller from the server. This is necessary for two reasons: 1.) the progress that is visualized by the pie charts and the timeline visualization have to be updated and 2.) the task might have been completed so that the worklist needs to be updated. In the latter case new tasks might have been enabled if the completed task has rules to other tasks assigned. Concurrent assignment of attributes to tasks is possible since the mandatory attributes are not cached. Every time a task is enabled in the worklist the entire page with its attributes is requested from the server.

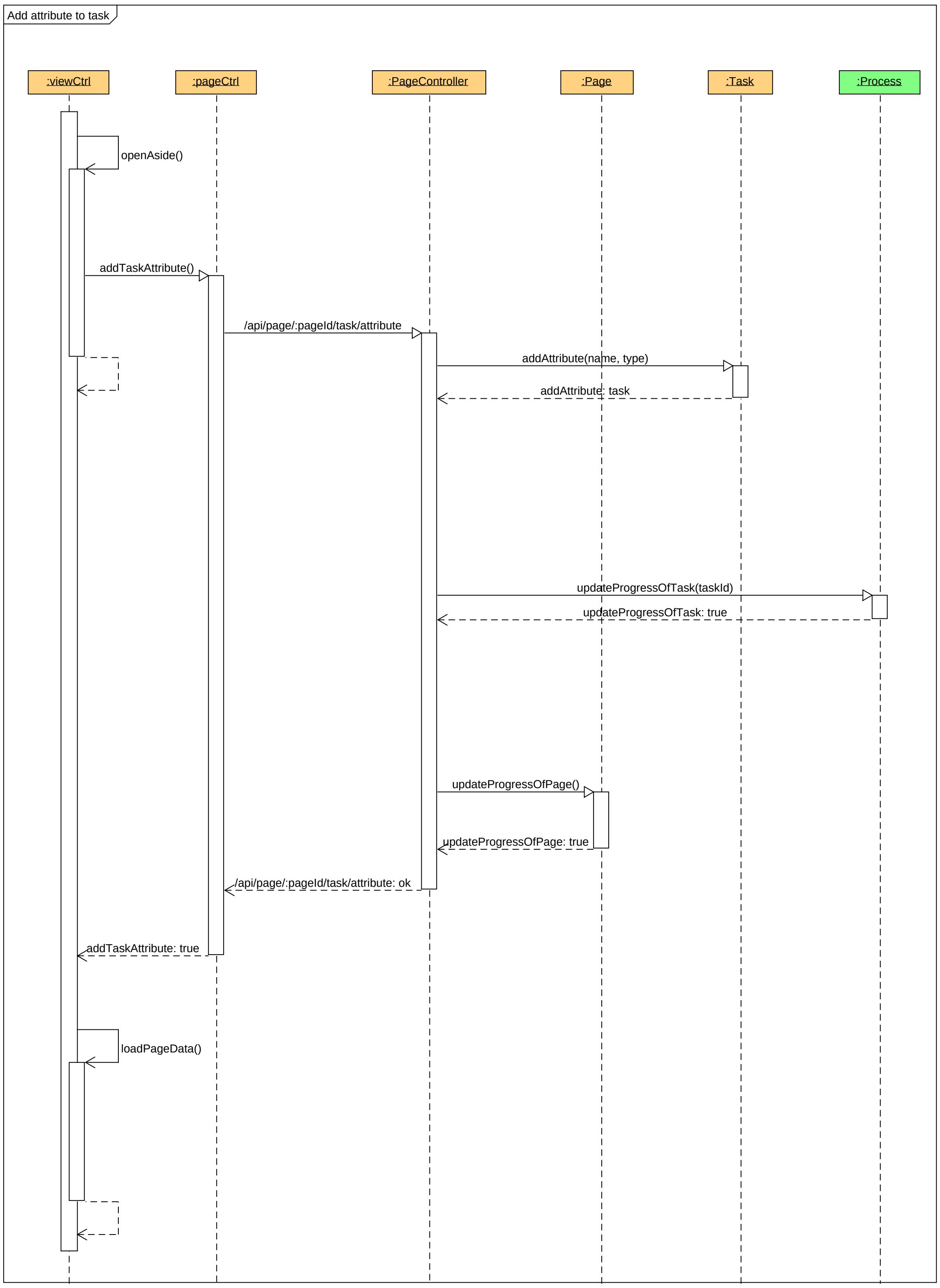


Figure 4.10.: Sequence diagram for the assignment of attributes to tasks on a wiki page

4.1.6. Progress Computation

The computation of the progress for wiki pages is an important part of the implementation. Most of the actions that users can perform in the system have an impact on the progress of wiki pages and tasks, e.g., entering attributes values, assigning attributes to tasks, or creating new tasks. After these actions the page controller automatically triggers the update for progress of the page. Figure 4.11 illustrates the sequence diagram with the required steps to update the progress of a wiki page. The process starts with the invocation of the *updateProgressOfPage* method in the page object. The page object starts a loop for all open tasks that are assigned on this page, i.e., completed tasks are omitted in the computation since their metadata is not necessary for the computation of the progress. For these tasks the method *findMetaData* is invoked to determine the values for the current progress. Based on the number of all tasks that are assigned to the page and the values for the progress of open tasks, the page object computes the current progress of the wiki page. After the new progress is computed the page object is persisted in the database, so that simple get request of wiki pages don't have to compute the progress.

The method updateProgressOfPage invokes another method in the page object called *updateProgressOfParents*. Due to the hierarchical organization of tasks, changes on a lower level have to be propagated to the root of the tree structure. The updateProgressOfParents method consists of another loop that iterates through all *parentIds* that are stored as list in every page object. For every parent page the list of relevant tasks is determined first. A relevant task has a mandatory attribute assigned that references the current page object, i.e., relevant tasks are influenced by the changing progress of the current page object. We also include already completed tasks of parent pages as relevant tasks. Thereby, an already completed parent task might be enabled again in case subtasks are enabled again. Enabling completed tasks is possible by performing the redo function in the worklist of the wiki page. The progress for these relevant tasks is updated with the method *updateProgressOfTask* by the process object in another loop. This method computes the task progress based on the formula presented in Section 3.2.3. During this update of the progress the state of the tasks is updated as well, i.e., inconsistent and overdue tasks are identified.

The inner loop is closed after the progress for relevant tasks of the parent pages are updated. The method continues with a recursive call of the *updateProgressOfPage* function. This method is invoked for all parent pages of the current page object, i.e., the update of the progress for pages and their tasks incrementally grows to the root page of the tree structure. After the recursive updateProgresOfParents method terminates, the enclosing updateProgressOfPage method returns true for the page controller. The invocation of the updateProgressOfPage method is used sparingly to avoid unnecessary computations. The latest progress for wiki pages and tasks is stored after every computation in a separate attribute (cf. Section 4.1.4). Therefore, it is not necessary for read operations that have no impact on the progress of tasks to invoke the udpateProgressOfPage method. With this approach the progress always reflects the current state of work that is stored in the system. Another opportunity would be to propagate the progress computation only at a certain moment in time, e.g., through nightly updates of the progress.

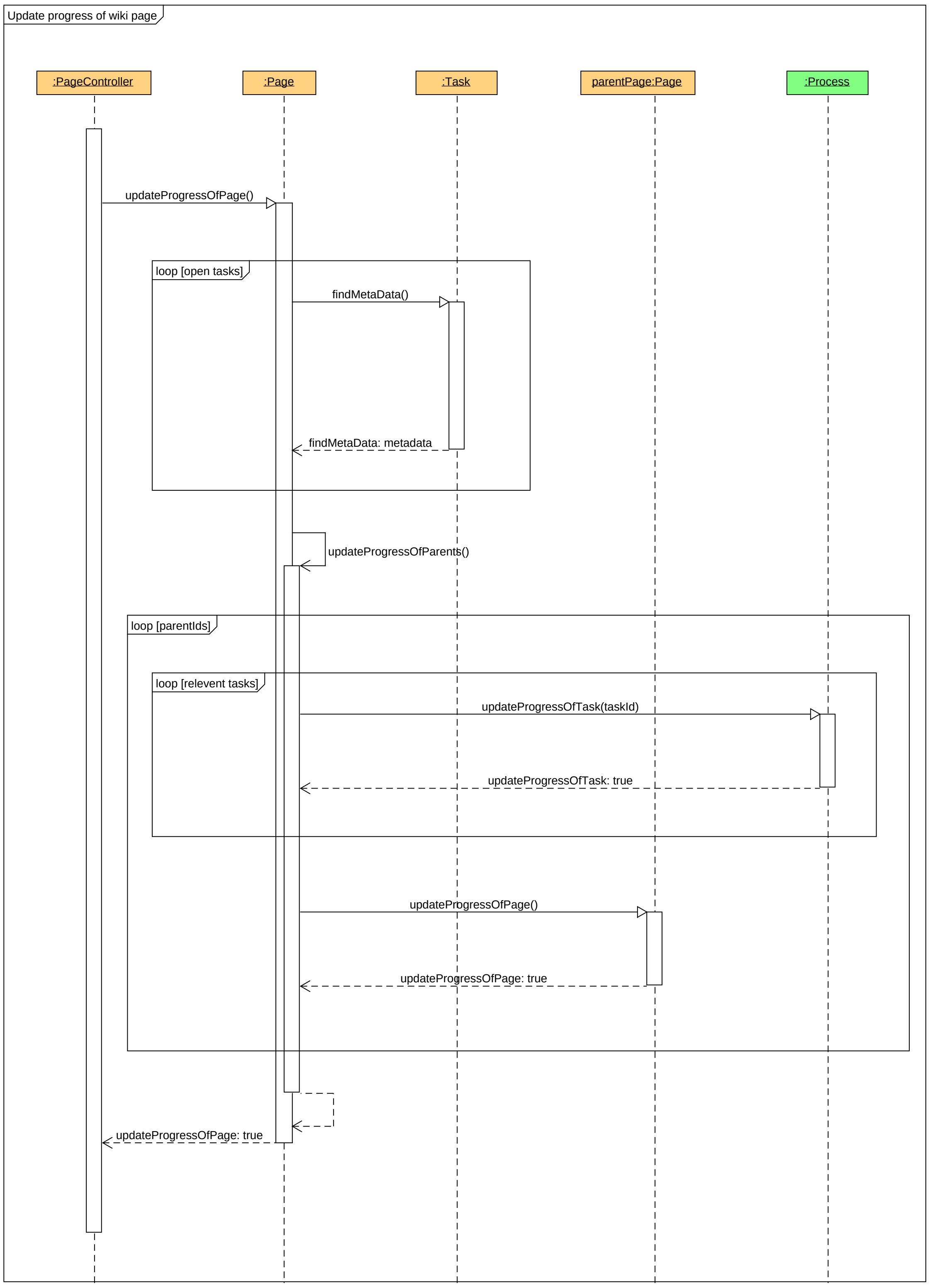


Figure 4.11.: Sequence diagram for the update of the progress of a wiki page

4.1.7. Generation of the User Profile

Users are able to the see profiles of other users in the system to get a shared understanding of their tasks and skills. The required operations to get a users' profile are illustrated in the sequence diagram in Figure 4.12. In the front-end the AngularJS controller *profileCtrl* creates a http GET request for the profile of a user with the *userId*. This request is routed to the method *profile* in the *UserController*. In the first step the UserController requests all tasks that have been created in the system using the *findAll* method. This is necessary because three subsequent methods require this information and requesting only tasks related to the user is not sufficient to determine the statistics that are shown in the profile. The first method *getClosedTasks* gathers all tasks that have been completed by the user. This includes tasks that are delegated and completed by the user. Every task is returned with the entire data structure of the Task and MetaData class. Similarly, the second method *getOpenTasks* returns all uncompleted tasks that are related to the user. The results for these two methods are used to show the open and closed tasks for the user in the profile with the information about the current progress. In Section 3.3.9 the user interface for the personal worklist is introduced as part of the profile.

The third method that is invoked to get the user profile is called *getUserRank*. This method is necessary for the rating feature that is introduced in Section 3.3.8. It counts the number of completed tasks for the user and divides them with the number of all tasks that are completed by the other users in the system. The resulting percentage is shown as ranking in the profile of the user, e.g., the user has completed more tasks than 100% of the other users (cf. Figure 3.15). In future work this method could take other factors for the computation of the ranking into account, e.g., some tasks have more weight depending on their expertises. Another sensible opportunity could be to apply the computation of the ranking only to a certain timeframe, e.g., only consider the ranking for one month. In case the ranking is computed for the entire lifespan of the application, new users that join the community might be deterred since a few experienced users would dominate the ranking. For new users it would be very difficult to achieve a higher ranking and this could be avoided with smaller timeframes for the ranking. The method returns to the profileCtrl with the data structure of the user profile.

Another http GET request is created by the profileCtrl with the method *expertiseChart*. This method creates the visualization of the expertises shown in Figure 3.15. We decided to split the http request for the profile and the expertises because we plan to use both requests separately in the future, e.g., to match tasks to users based on their expertises. The request for the expertises is routed to the UserController that invokes the method *getFinishedTasks* of the user. All finished tasks with the associated MetaData objects are returned to the UserController. For every completed task the expertise in the MetaData are collected in a loop. We only consider completed tasks in the computation of the expertises. For the subsequent visualization expertises with a percentage that is below 5% are removed since they are not readable in the skills chart. The filtered expertises are returned to the profileCtrl that invokes the *Highchart* visualization[12]. Highcharts is a JavaScript library for interactive visualizations that provides several visualization types, whereas the expertises are shown in a pie chart. The expertises can be selected in the pie chart to filter the tasks with these expertises.

[12]`http://www.highcharts.com/`, last accessed on: 2015-06-20

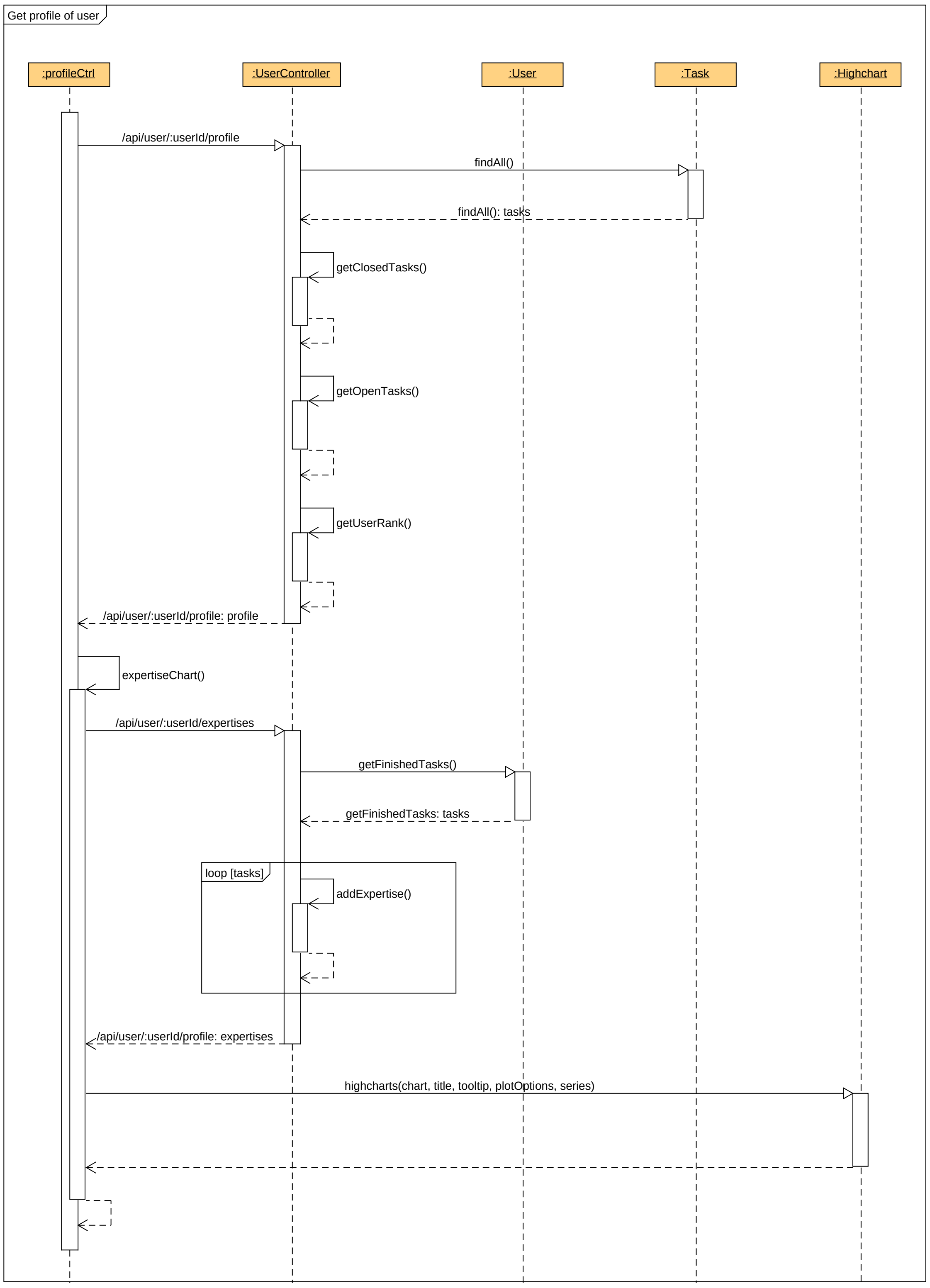


Figure 4.12.: Sequence diagram for the generation of the profile and expertises for a user

4.1.8. Loading of History Entries in the Social Feed

The social feed is an important part of the user interface since it is shown as landing page for the users and it aggregates all activities in the system. Figure 4.13 illustrates the sequence diagram for the loading of history entries in the social feed. In the front-end the AngularJS controller *feedCtrl* invokes the method *getNextEntries* several times through a *callback* function. The method is invoked for every filter that can be applied in the social feed, i.e., data, tasks and discussions. The method is invoked separately because only history entries for three pages that are visible in the user interface are gathered from the server to avoid performance issues. After the user navigates to the third page with the pagination of the social feed that is located at the bottom, the feedCtrl automatically gathers the subsequent three pages with history entries. In case three pages for all filters in the social feed are gathered, it might be that not enough history entries are available. This might appear for instance if the last three pages of the social feed only consist of task history entries and the user applies the filter for discussions or data. The getNextEntries method creates a http GET request that is routed to the *UserController*.

The UserController uses the method *findAll* of the *PageHistory* companion object with the *userId*. The userId is provided as parameter to update the set with the userIds that have seen the entries in the PageHistory. The method *updateSeenUsers* updates this set for every instance of the HistoryEntry class, e.g., DiscussionEntry, TaskHistoryEntry, or AttributeHistoryEntry. This set is used to update the badge with the number of new history entries that is shown in the navigation bar, which is computed for every user individually. Therefore, every visit of the social feed automatically resets the badge in the navigation bar. After the update of the seen users the PageHistory returns all history entries to the UserController. These history entries are filtered with the *filterInput* parameter that is determined by the user in the social feed, i.e., data, discussions and tasks. Users can select more than one filter with arbitrary combinations of them. The filtered history entries are sorted by their date and returned to the feedCtrl. The feedCtrl assigns the new history entries to the model and this triggers the update of the social feed in the view. Depending on the specified filters the feedCtrl invokes the getNextEntries method again using the callback function or continues with the remaining operations.

Every history entry in the social feed can be commented by other users in the system, e.g., other users can comment the history event that is created for the completion of a task. New comments for events are immediately shown in the social feed without a reload of the page in the browser. With this feature it is possible to directly respond to comments of other users in realtime. For this purpose the feedCtrl invokes the method *listen* to create a request that is routed to the UserController. In the UserController a new event stream is created for the user. The feedCtrl adds this event stream with the method *addEventListener* to the social feed. In the current implementation it is not possible to restrict history entries to certain groups or users, i.e., every users has the same history entries in the social feed. Although the history entries are visible for everyone in the system, access to the attributes remains restricted. In future work visibility of history entries could be determined by the access rights for the attributes. Another sensible extension would be to limit the visibility of discussions entries for certain users or groups.

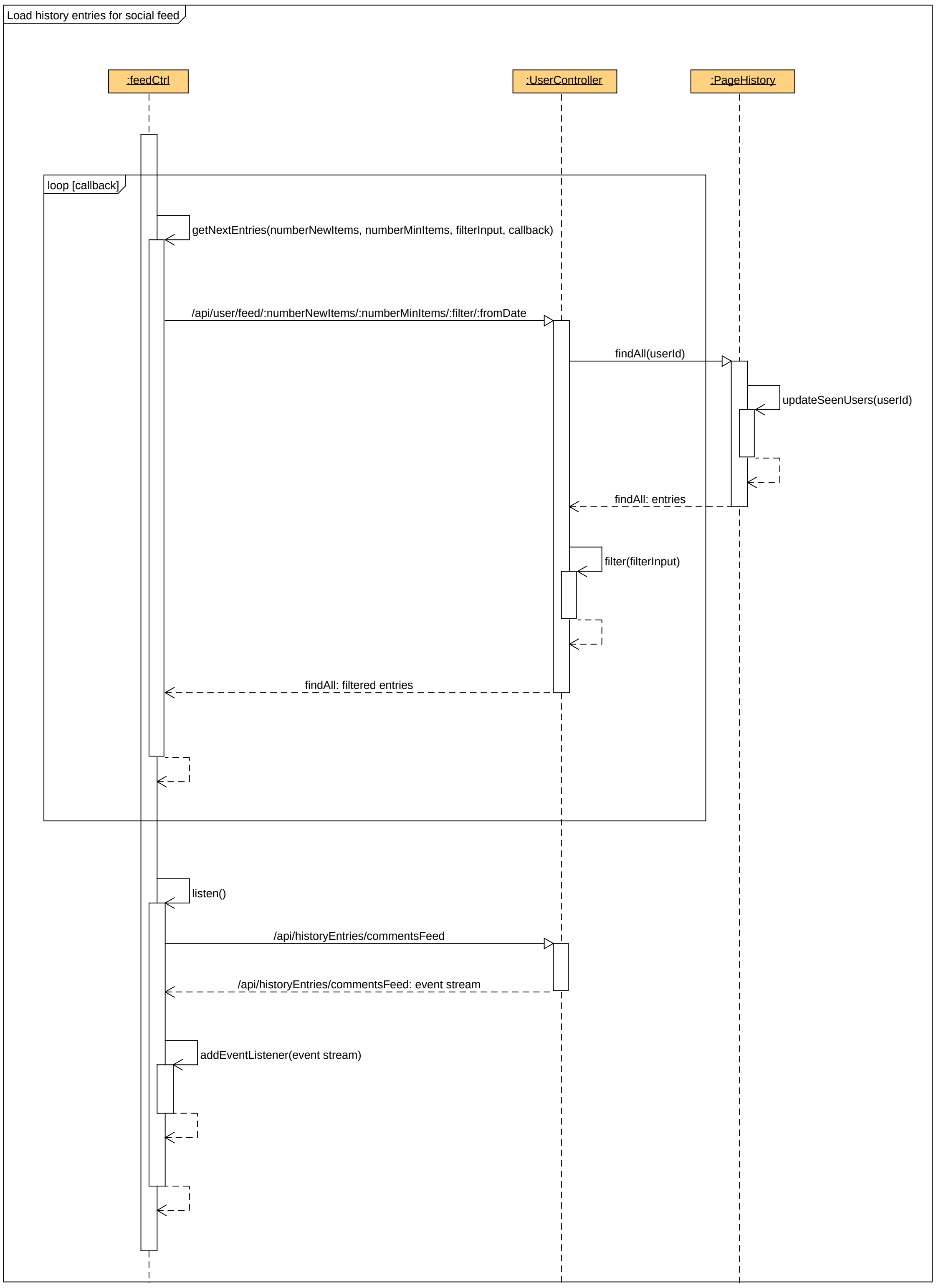

Load history entries for social feed
:feedCtrl
:UserController
:PageHistory
loop [callback]
getNextEntries(numberNewItems, numberMinItems, filterInput, callback)
/api/user/feed/:numberNewItems/:numberMinItems/:filter/:fromDate
findAll(userId)
updateSeenUsers(userId)
findAll: entries
filter(filterInput)
findAll: filtered entries
listen()
/api/historyEntries/commentsFeed
/api/historyEntries/commentsFeed: event stream
addEventListener(event stream)

Figure 4.13.: Sequence diagram with required steps to load history entries for the feed

4.2. Case Management Model and Notation Workbench

In this section the interactive CMMN workbench that has been developed in [Ge14] is explained in the detail. This CMMN workbench is not visible for authors and can be shown in a separate user interface for tailors. In contrast to the features for the collaborative structuring of KiPs introduced in Section 3.3, the workbench requires knowledge about process modeling with CMMN. Nevertheless, it should be possible for tailors without programming skills to maintain work templates with this editor. Tailors that work with this workbench can be authors at the same time in our approach. We envision that selected authors for work plans can become tailors through some training on the most important concepts of CMMN. For this purpose we decided to implement a subset of the CMMN language that includes the most relevant elements that are necessary to specify KiPs. In Section 3.1.2, we introduced two possibilities for the evolution of KiPs in our approach that can be combined with each other. In the first possibility the structure of KiPs emerges through contributions of authors and tailors are mainly responsible for the maintenance of the work templates. In the second possibility tailors predefine skeletons or blueprints for KiPs that are refined by authors during the execution. In either possibility tailors can restrict the freedom of authors with rules that impose certain dependencies in the KiPs, i.e., validation steps always have to be performed by a certain role in the work plan. This approach supports the different degrees of structure for KiPs that are necessary for the variety of application scenarios in this area. Both possibilities are supported in our implementation of the workbench and described in the subsequent sections.

4.2.1. Introduction to JointJS

The technical foundation for the CMMN workbench is the JavaScript diagramming library JointJS[13]. The library provides generic functionalities for the visualization and interaction with diagrams and graphs in the browser. The library is implemented with a simple model-view-controller (MVC) architecture. The model consists of graphs that contain elements and links with presentation attributes. In the view all graphical elements are rendered with Scalable Vector Graphics (SVG). Main reason for the usage of the library in this thesis is its ability to develop interactive diagramming tools. Several diagram types are already readily available in the framework, e.g., BPMN, UML and finite state machines. These diagrams are serialized and deserialized to JavaScript Object Notation (JSON) format, which makes it suitable for the integration into our API that is based on JSON. Due to the recent publication of CMMN the library provides no built-in support at the time when this thesis is being written. Therefore, we implemented a new diagramming type in JointJS that is based on a subset of the visual notation of CMMN. The developer of the library additionally provides a complete diagramming toolkit for commercial applications. In this thesis only the open source core of the library without the commercial diagramming toolkit is used. One of the limitations of the library is that it provides no built-in support for the automated layouting of graphs. For this purpose another implementation is available that integrates the layouting based on Dagre[14] in JointJS[15]. Dagre is a directed graph layout engine that can be used in JavaScript.

[13]`http://www.jointjs.com`, last accessed on: 2015-06-26

[14]`https://github.com/cpettitt/dagre`, last accessed on: 2015-06-26

[15]`http://www.daviddurman.com/automatic-graph-layout-with-jointjs-and-dagre.html`, last accessed on: 2015-06-26

4.2.2. Measurement of the Method Complexity

An important aspect about process modeling languages that we consider in this thesis is the issue of understandability [Fa09]. With every iteration of new process modeling standards, these languages tend to become more and more complex, e.g., BPMN 2.0 [Al10]. In practice not all constructs that are provided by these process modeling languages are useful or even necessary for business analysts. In order to address this issue some authors proposed the identification of subsets for BPMN (e.g., Zur Muehlen et al. in [ZMR08]). A challenging issue for this problem is the comparison of different subsets of process modeling languages to determine which subset performs better in terms of understandability. One possibility to measure an indicator for the understandability of process modeling languages is the metamodel based complexity presented by Rossi and Brinkkemper in [RB96]. Equation 4.1 describes how the complexity $C'(M)$ of a process modeling language is computed based on its metamodel. In the equation O_M captures the objects that are defined in the metamodel of M. R_M contains the relationships that are defined in M between the objects. P_M contains the properties that are defined in the objects. The value of this equation might be an indicator for the understandability of a process modeling language, i.e., higher values of $C'(M)$ might imply that the language is more difficult to understand for modelers. In our approach authors already work with a very limited set of lightweight structuring concepts, e.g., tasks, attributes, types and metadata for tasks. For tailors we also think that it is important to reduce the complexity of the process modeling language to make the maintenance of work templates easier.

$$C'(M) = \sqrt{n(O_M)^2 + n(R_M)^2 + n(P_M)^2} \tag{4.1}$$

Within a master's thesis [Ge14] this computation has been applied on the metamodel of the solution presented in this thesis. The result for the cumulative method complexity of the proposed modeling language in this thesis is described in Equation 4.2. The values for the objects, relationships and properties are determined based on the data model introduced in Section 4.1.4. Only concepts that are relevant for the specification of KiPs are considered in the computation, i.e., technical concepts that are only necessary for implementation reasons are omitted. The objects contain the structuring concepts of the work plans and templates since tailors need to be able to distinguish between them. Furthermore, the different types of attributes are incorporated as individual objects, e.g., boolean, string, integer, date and page values. The relationships contain two values for the rules that can be defined between tasks and stages. The other two values capture the relationships from pages to attributes and the type. Properties consist of the metadata that can be specified for tasks with the progress, expertise, start and end dates. The remaining properties are related to names and values for the objects. The overall cumulative complexity $C'(M)$ for our process model is $34,29$. In Table 4.1 the cumulative complexity of other process modeling languages is summarized based on findings of Marin et al. [MLVDP14]. BPMN is represented with its entire specification and subsets that have been proposed by other researchers. Compared with existing approaches the solution presented in this thesis has the lowest complexity except for event-driven process chain (EPC) and UML activity diagrams.

$$C'(M) = \sqrt{n(O_M)^2 + n(R_M)^2 + n(P_M)^2} = \sqrt{22^2 + 4^2 + 26^2} = 34,29 \tag{4.2}$$

Method	Obj	Rel	Pro	Cumulative Complexity
BPMN 1.2	90	6	143	169.07
BPMN 1.2 DoD	59	4	112	126.65
BPMN 1.2 Case Study	36	5	81	88.78
BPMN 1.2 Frequent Use	21	4	59	62.75
CMMN 1.0	39	4	28	48.18
DARWIN	22	4	26	34.29
EPC	15	5	11	19.26
UML Activity Diagrams	8	5	6	11.18

Table 4.1.: Comparison of the complexity for different modeling languages [Ge14]

The results of the cumulative complexity measurement for the different process modeling languages is illustrated in Figure 4.14. The height of the vertical lines describes the result of Equation 4.1 for these methods. The full specification of BPMN 1.2 has the highest complexity of all methods that are investigated. CMMN has a much lower complexity compared to BPMN and all of its subsets, which is already one advantage of CMMN for the support of KiPs according to [MLVDP14]. The process model proposed in this thesis reduces the complexity of CMMN even more since only a subset of the proposed objects and properties are used. Only EPC's and UML activity diagrams have a lower complexity since these languages provide fewer properties. In future work the process model for work templates might become more complex in case more functionalities are required in practical application scenarios. For the lightweight structuring concepts in the work plans we expect much less changes in the future, i.e., the concepts introduced in Section 3.2 should remain rather stable.

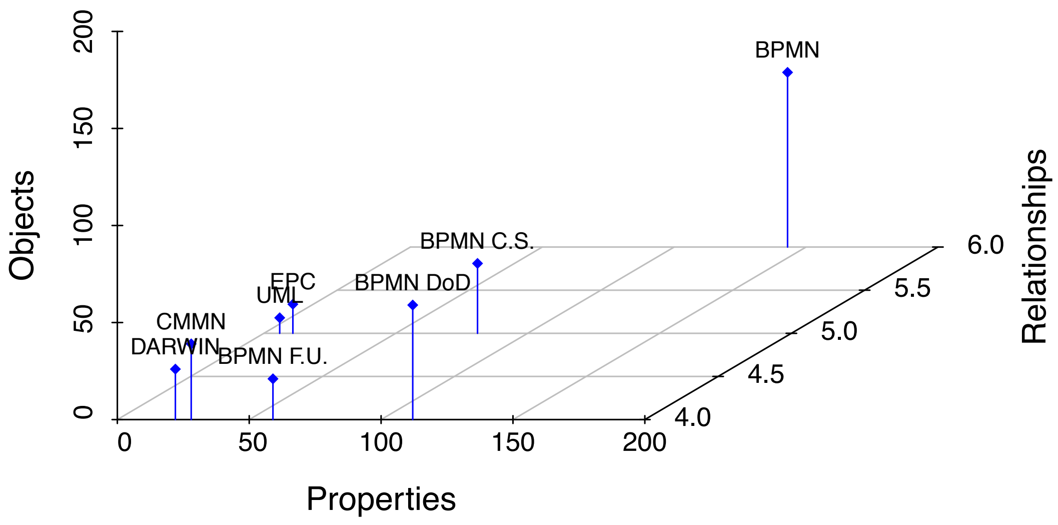


Figure 4.14.: Visualization with the cumulative complexity for our approach [Ge14]

4.2.3. Interaction with Elements

In this section the basic interactions with elements in the CMMN workbench are described. In the current implementation the CMMN workbench can be accessed by clicking on the name of a work template. In future work the CMMN workbench could be migrated to a separate user interface, e.g., another web interface that can be accessed with a browser or a client that is based on eclipse[16] or Microsoft Visio[17]. In the CMMN workbench tailors can maintain the process structure for work templates. Figure 4.15 shows a screenshot with an excerpt of the CMMN workbench. The workbench contains the tasks that were already used in the previous examples. Above the CMMN diagram the TypeDefinition *Page* of this work template is shown. Two buttons are displayed after tailors enter the diagram with their mouse cursor. These two buttons can be used to create a *new stage* or *task*. After clicking on one of these buttons a new window opens that can be used to enter the required information for the new stage or task. Below the buttons the diagram contains the tasks *A* to *H* that are created in the sample. In this sample the process contains no stages and there are no dependencies defined with rules. The tasks contain an icon that is used to denote so called blocking human tasks in CMMN, i.e., these tasks are waiting until the work is completed. The position of the elements in the CMMN workbench can be changed and the coordinates are stored in the database.

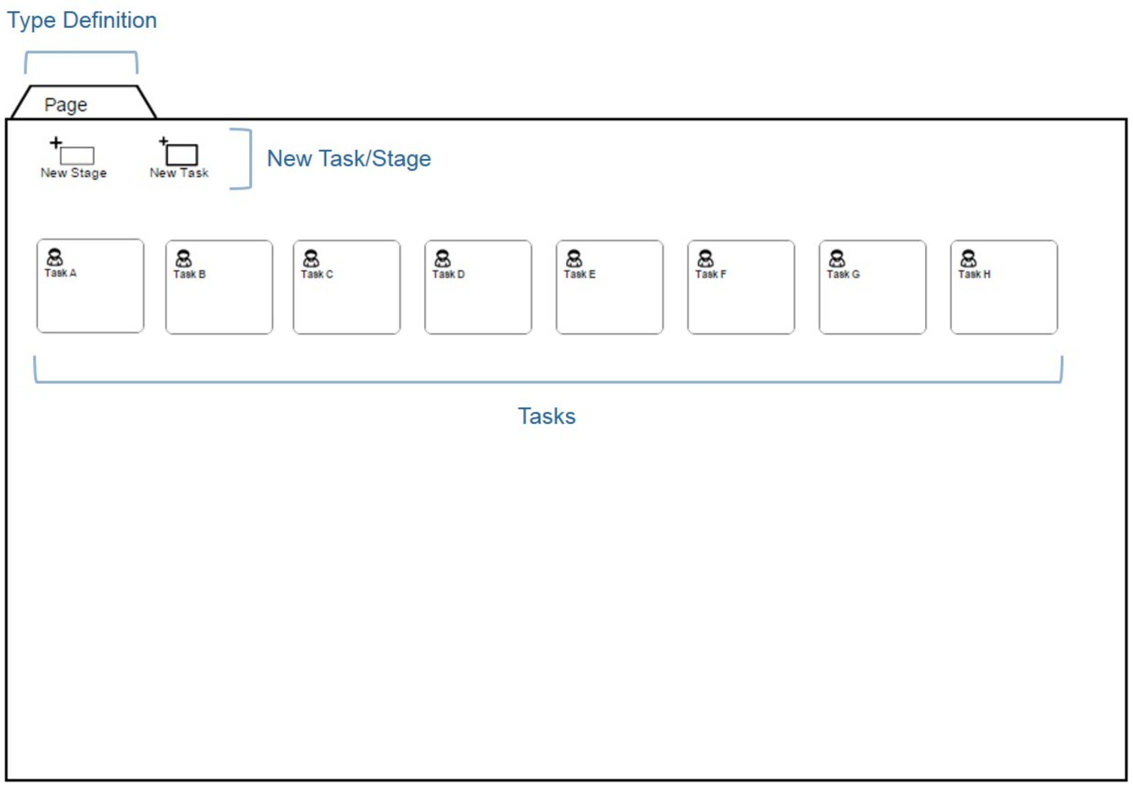


Figure 4.15.: Excerpt of the CMMN workbench that contains the previously used samples

[16]https://eclipse.org/, last accessed on: 2015-06-29
[17]https://products.office.com/de-de/Visio/, last accessed on: 2015-06-29

The interaction with tasks is illustrated in Figure 4.16 with the sample *Task C*. After hovering of the task element an additional button appears similar to the creation of a new task or stage described in the previous example. This button shows an arrow with a plus sign that can be used to create a new rule for the selected task. After clicking on this button a new window appears that is used to select which task or stage should be associated with the new rule. Since Task C has no rules associated in this example, it is shown immediately after the work plan is created. The name of the task is automatically shortened in case it exceeds the available space of the shape.

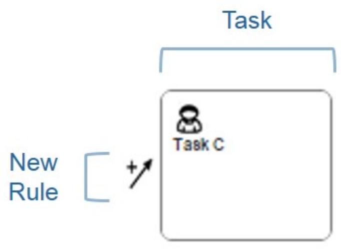
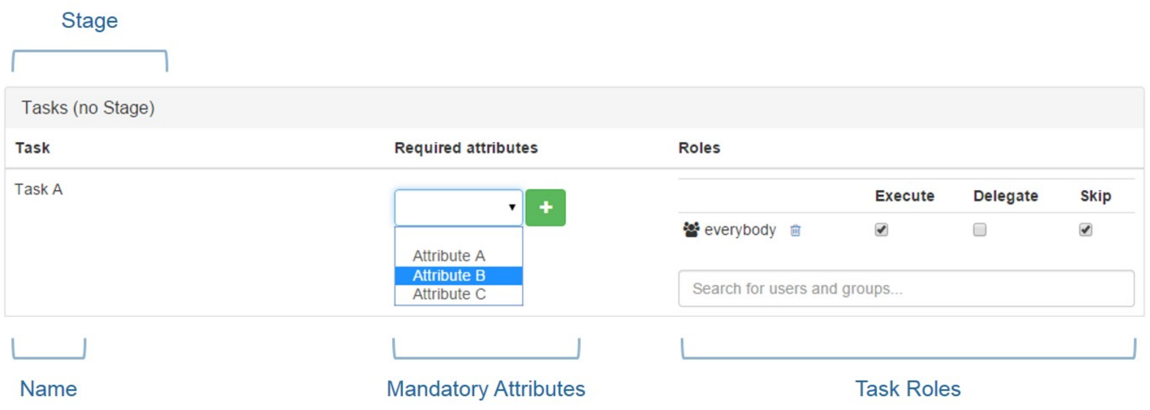


Figure 4.16.: Task interaction

CMMN only provides visual elements that are directly related to the process structure. Although the standard contains a generic information model, it provides no solution how this information model can be linked to the process structure in the notation. Therefore, it is necessary that the runtime environment provides content management services for CMMN, e.g., based on the Content Management Interoperability Services [MB15]. Figure 4.17 shows how the properties for the sample *Task A* can be defined in the workbench. In the first column of the table the name of the task is shown. Mandatory attributes for the task in the work template can be defined in the second column. In the work plan authors can perform this operation with drag and drop for one instance of the template. In the third column the roles for the task in the work template are defined. New roles can be added by writing the name of the group or user in the input field. In the example Task A can be executed and skipped by every user that is registered in the system. Similarly, all tasks in the work template have their own properties in the workbench that are categorized by the stage of the task.



Tasks (no Stage)

Task	Required attributes	Roles	Execute	Delegate	Skip
Task A		everybody	✓	☐	✓

Figure 4.17.: Screenshot of the properties for one task in the CMMN workbench

In our approach work plans might have no restrictions at all to support unstructured processes in which this is not desired, e.g., science processes described in [Gi15b]. Another possibility could be that rules are unknown due to a low maturity of the work template and they need to emerge first. We delineate two cases in which the definition of rules is necessary. After several iterations of the same work plan recurring patterns for dependencies might arise, e.g., some traces of tasks are always executed in the same order (case 1). In other application scenarios dependencies might be required to avoid undesired behavior (case 2). Figure 4.18 shows an example for a simple rule between two tasks in the CMMN workbench. In this thesis we refer to producer and consumer tasks that are associated with rules. A producer task needs to be completed before a consumer task can be enabled. In the example in Figure 4.18 *Task D* is a consumer of producer *Task C*. After the work template is instantiated the consumer tasks are not shown in the task representation and the timeline. During the execution of the work plan consumer tasks are enabled once their associated producer tasks are completed. In future work the number or names of consumer tasks that are enabled after another task is finished could be shown in the work plan.

Main advantage of rules in case 1 is that authors are not overwhelmed with too many enabled tasks. This might happen if work plans have a large number of tasks and it becomes difficult for users to select suitable next steps. With rules the number of enabled tasks can be reduced since consumer tasks that can only be started at a later stage in the work plan are not shown. In case 2, certain traces of tasks in the work plan have to be ensured with the rules, e.g., if there are compliance requirements that should be adhered to. In addition to this simple producer and consumer dependency shown in Figure 4.18, it is possible to define several variations of this rule. It is possible to define more than one consumer task with rules, i.e., all associated consumer tasks are enabled at once. Furthermore it is possible to define multiple producer tasks for one rule. Two variations are possible in case two or more producers are associated with one consumer. The rule can have either an AND or an OR logic associated. With the AND logic all producer tasks have to be completed before the consumer is enabled. The OR logic only requires one of the producers to be completed before the consumer is enabled. Despite its benefits rules need to be carefully introduced since they limit the freedom of knowledge workers. In our approach tailors are responsible for the maintenance of rules and they are able to remove them again from the process structure if necessary.

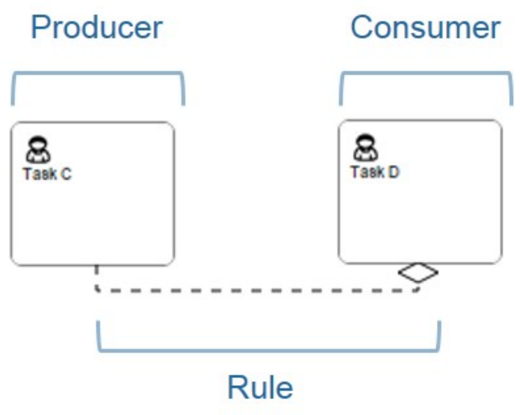


Figure 4.18.: Example rule for two tasks

Similar to data-centric business processes that are described with state machines, stages in CMMN represent the behavior of cases. In the CMMN specification stages are also referred to as episodes of a case [Ob14]. Main advantage compared to an approach that is based on state machines is the higher flexibility at runtime since CMMN uses a declarative model. In our approach authors are not able to define stages on work plans since this would dramatically increase the level of complexity for end-users. Only tailors are able to define stages for work templates using the button on the top of the diagram in the CMMN workbench. In line with our evolutionary model for work templates there are two possibilities when the stages can be defined in the beginning. In the first possibility tailors can predefine stages as a blueprint for the work plans that are executed by authors. In the second possibility tailors create stages based on the tasks that are defined by authors. After several tasks emerged in the work template, it might be easier to define the behavior with stages. In either possibility tailors are able to maintain the emerging process structure once the work plans are used by authors.

Figure 4.19 shows a stage after its initial creation in the CMMN workbench. Tailors can move already existing tasks that are related into the stage using the drag and drop functionality. Tasks that are grouped within one stage are usually processed together. After hovering over the stage available operations that can be performed are shown above and next to the stage. Tailors can also create new tasks directly within the stage using the *new task* button next to the stage. The only difference to the previously introduced new task operation is that the task is automatically assigned to the stage. Similar to the task it is possible to associate stages with rules using the *new rule* button. Rules can be associated to other stages or tasks. During the creation of a rule for tasks it is also possible to assign stages. Stages can also be deleted again using the *delete stage* operation. Before deleting a stage is possible all tasks within this stage have to be moved outside again.

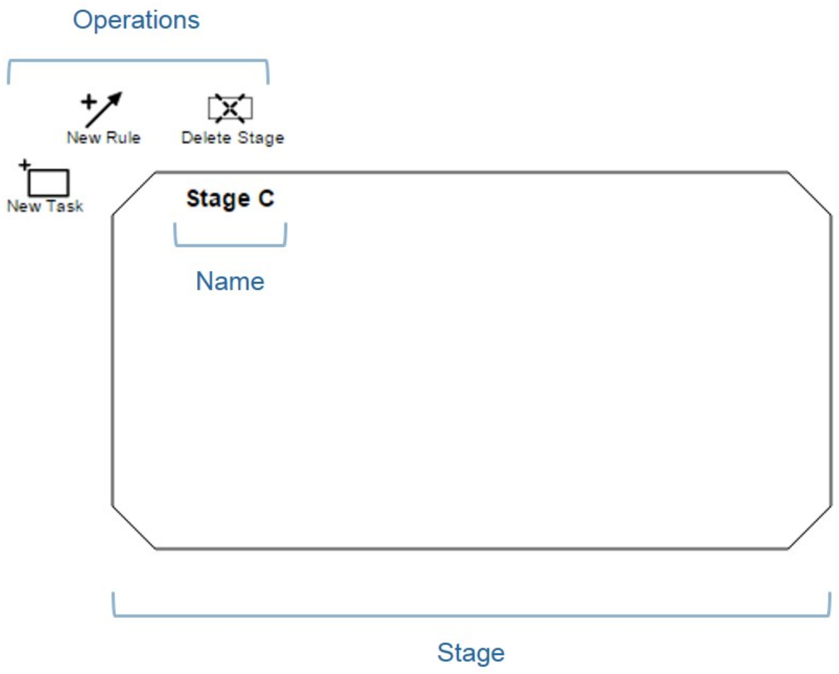


Figure 4.19.: Available operations after hovering over a stage

Figure 4.20 shows an example rule that associates a producer task with a consumer stage. In this example *Stage C* contains the two tasks *C1* and *C2*. These two tasks have a different icon compared to the other tasks that were presented before. Task C1 and C2 have a folder icon since they call another work plan. In the work plan this is represented with a mandatory attribute that references another wiki page. Regarding the rules between tasks it makes no difference which kind of task is associated. After the producer *Task A* is completed the consumer *Stage C* is automatically enabled. In case the stage is enabled both tasks within this stage are enabled as well. It is also possible to define stages as producers whereas the stage is finished after all tasks within this stage are completed. Furthermore, rules associated with stages can have multiple producers or consumers with an AND or OR logic. The behavior is exactly the same as described previously for the tasks. Rules can also be applied on tasks within a stage, e.g., Task C1 and C2 could again be associated with a rule. In this case only Task C1 would be enabled after Stage C is entered. Even though tasks are moved to a stage it is still possible to directly associate tasks with rules, i.e., Task A could be associated with Task C1 or C2.

Regarding the case studies conducted in this thesis the CMMN workbench provided sufficient expressiveness to implement the processes. In future work the expressiveness of the CMMN workbench could be extended since the implementation of the entire specification exceeds the scope of this thesis. This includes the adoption of the CMMN exchange format to allow interoperability with other case management tools. In a next step the CMMN element called process task could be implemented in the CMMN workbench. With the process task CMMN allows the invocation of a structured business process, e.g., to integrate processes described in BPMN with the solution presented in this thesis. In future work it could be valuable to extend the CMMN workbench with collaboration features. Although the number of tailors working with the CMMN workbench will be much lower compared to the number of authors in most cases, an improved synchronous and asynchronous collaboration might improve the quality of the work templates.

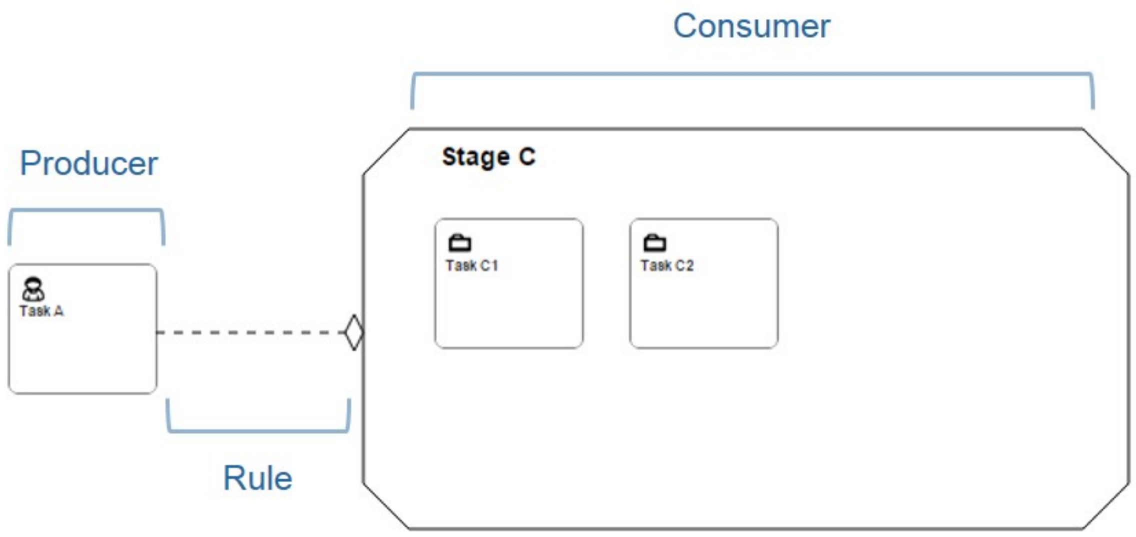


Figure 4.20.: Example rule associating producer task with consumer stage

4.2.4. Suggestion of Task Definitions

In the previous examples the creation of new elements in the CMMN workbench was introduced. In this case tailors can define the process structure based on their own knowledge or experience about the KiP that needs to be supported. This can be helpful to create an initial blueprint of the work template, so that it is not necessary for authors to start from the scratch with an empty wiki page if this knowledge is available. As described in Section 3.1, work templates in our approach emerge through the contributions of authors that maintain lightweight structuring concepts at runtime. The second main purpose of the CMMN workbench is that tailors can analyze these contributions in order to reveal recurring patterns that can be generalized in work templates. Figure 4.21 shows an excerpt of the CMMN workbench with *Task I* that is created by an author. Task I is not defined in the work template and represented with dashed lines. After tailors hover with their mouse cursor over Task I, the workbench automatically displays usage statistics at the top of the diagram. In the Hybrid Wiki a similar functionality was proposed to analyze the usage of attributes on wiki pages [Ne12]. With this simple statistics tailors can be supported with the decision whether *Task I* should be defined in the work template. Similarly, other usage statistics could be implemented to analyze how often a task has been skipped by authors on work plans. Tasks that are skipped very often might be suitable candidates that can be deleted from the work template.

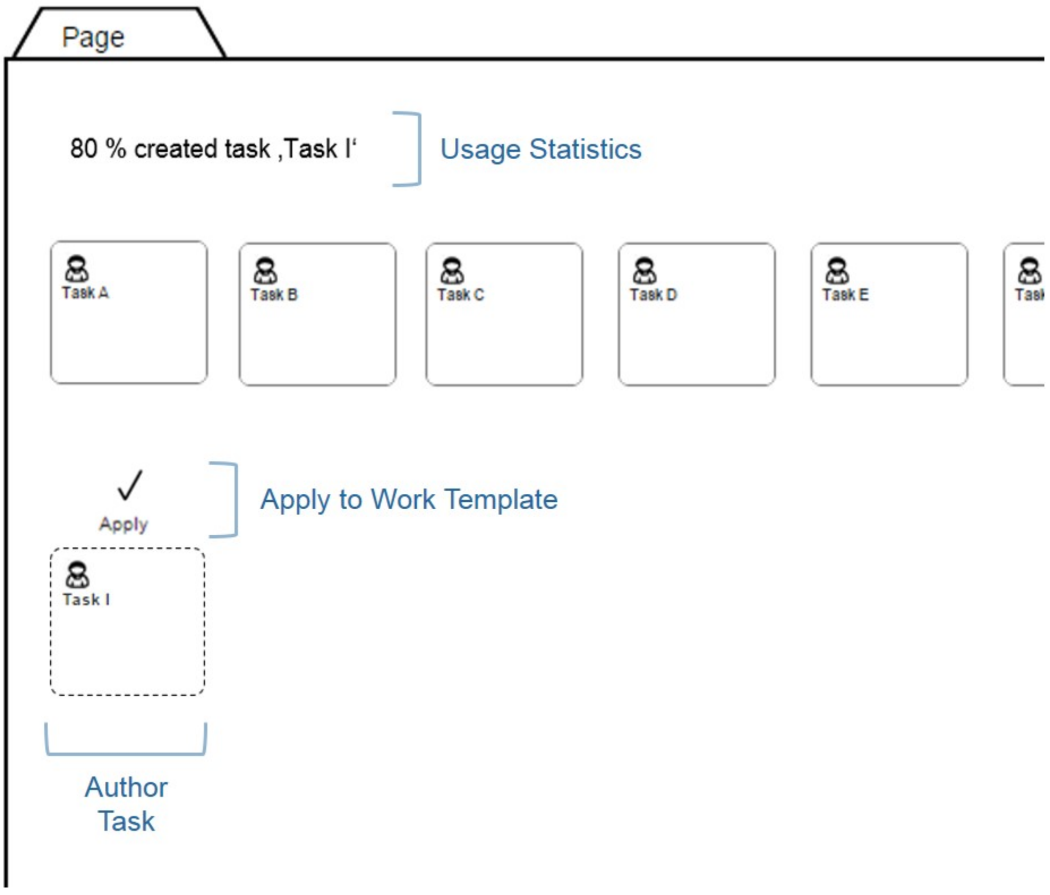


Figure 4.21.: Suggestion of a task based on usage statistics for the work template Page

Above Task I a new button appears after tailors hover with their mouse over the task. With this button tailors can apply the task to the work template of the Page, i.e., a new TaskDefinition for Task I is created for the TypeDefinition Page. After the TaskDefinition is created every new instance of the Page work template has this task defined. This mechanism allows the definition of work templates based on the lightweight structuring concepts provided by authors. In the current implementation all tasks that are created by authors are shown in the CMMN workbench with dashed lines. In future work the number of tasks that are suggested to tailors needs to be reduced if too many tasks are created by authors, e.g., based on the usage statistics. More sophisticated usage statistics and analysis would be valuable for tailors as well. A recent example for such more sophisticated analysis in the area of case management can be found in [Yi15]. Clustering of tasks could be helpful to identify similar tasks that have different names. The similarity of tasks could be computed based on their assigned attributes or other tasks that are executed before. This cluster could be proposed to tailors as one coherent TaskDefinition. The suggested tasks could also be shown as recommendations on work plans for authors. This could improve the consistent usage of the names that are used for tasks and attributes.

Another extension of the workbench would be the suggestion of rules based on the actually performed execution order of tasks in work plans. In case some tasks are always executed in the same order, the workbench could suggest rules between these tasks for tailors. An example that is based on the application of the apriori algorithm for process models defined with CMMN can be found in [SZJ13]. The workbench could provide an apply mechanism for suggested rules similar to the example for tasks shown in Figure 4.21. Main advantage of the CMMN workbench proposed in this thesis is that these analysis mechanisms can be applied based on the structure provided by authors. Without this structure the analysis could only be based on simple rather unstructured log files, which makes it much more difficult to reveal recurring patterns in the processes. The implementation of the CMMN workbench provided in this thesis is only a proof of concept and future iterations of the workbench need to be improved with more sophisticated analysis techniques to support tailors. This includes the application of mining algorithms and the visualization of usage statistics. Every element in the CMMN workbench could provide usage statistics to tailors, e.g., number of users skipping a task or average duration of tasks.

In addition to the structure provided by authors on work plans in this system other external information sources could be integrated with the CMMN workbench. Due to our generic information model a variety of different events could be stored and analyzed in our system. Potential information sources that can be analyzed are for example e-mail clients, enterprise wikis, or integrated development environments for software development. Many of these tools already provide means for tasks or related concepts to describe process elements that can be mapped to our model. This data can be imported with the methods exposed by the API that is introduced in Section 4.1.3. Based on this information the analysis of work plans can be used to reveal patterns within the incorporated external tools. The suggested work plans can be executed in Darwin or shown directly in the information sources. Another possibility would be to extend existing information systems with the lightweight structuring concepts introduced in this thesis. With this solution tasks with their metadata and their link to attributes can be directly created in these information systems.

4.3. Implementation for Mobile Devices

In this section important implementation aspects of the mobile solution are described. More detailed descriptions of the implementation are available in the master's thesis in [Ab15]. The implementation for mobile devices consists of a new user interface that is developed entirely new with Angular Material which is briefly introduced in Section 4.3.1. Just like the desktop interface the user interface for the mobile solution is based on the API that is described in Appendix A. Only one adaptation of the server is necessary to switch the templates for the front-end in case a mobile device is detected as requesting client. Some features that are available in the desktop client are disregarded for mobile devices, e.g., the assignment of attributes to tasks and the upload of files for attributes. These feature are either not relevant in this context or difficult to realize on mobile devices. Primary use cases for the mobile interface are the quick creation of tasks and the collaboration within the social feed. Completing tasks through the input of data for attributes is still mainly performed in the desktop interface. Furthermore, the mobile interface is only designed for authors and visitors. Features for the maintenance of work templates through authors are not implemented in this client. Therefore, the mobile solution is not replacing the desktop version but it allows the maintenance of already existing work plans.

In Section 4.3.2 the navigation for the mobile interface is described. Due to the smaller screen on mobile devices the navigation has been revised. The mobile navigation allows quick overview about tasks that are due in the near future and it provides access to the social feed as well as individual work plans. In contrast to the desktop version there are no information about work templates shown in the navigation structure. The infinite scrolling for the social feed in the mobile user interface is described in Section 4.3.3. It automatically reloads history entries after the user has scrolled to the end of the feed. The usability of the mobile interface is tested with real world users in an experiment. Some issues that were identified in this experiment are already considered in the current implementation. These issues and results of the usability test are summarized in Section 6.4 within the evaluation chapter.

4.3.1. Introduction to Angular Material

The mobile implementation is based on Angular Material[18], which is an early implementation of Material Design[19] in AngularJS. Main goal of material design is to provide a unified solution for visual, motion and interaction design. It is applicable across different devices and screen sizes, i.e., the desktop version of Darwin could be implemented in material design as well. It consists of a set of guidelines for the creation of minimal designs that are based on metaphors. Minimal design is important due to the high amount of mobile devices. Unnecessary elements in the user interface have to be omitted to avoid distraction of users and improve performance since lightweight pages have better loading times. Metaphors provide tactile reality to the user through visual cues that are grounded in reality, e.g., animations and shadow. The consistent use of these metaphors in combination with minimalistic design contributes to the intuitive usage of web based applications. Angular material provides an implementation of these guidelines including services and directives that can be customized.

[18]`https://material.angularjs.org`, last accessed on: 2015-08-20
[19]`https://www.google.com/design/spec/material-design`, last accessed on: 2015-08-20

4.3.2. Navigation

The navigation menu for the mobile interface is illustrated in Figure 4.22. Due to the smaller screen size and the different usage scenarios of the mobile interface the navigation has been adapted. This navigation menu can be reached at any time with the hamburger in the navigation bar. The main navigation is divided into three categories for news, tasks and data. The news category provides a link to the social feed. The category tasks contains two items for today and next 7 days. Today lists the tasks assigned to the logged in user that are due today. Next 7 days lists tasks that are due within the next week for the user. The data category is similar to the explorer in the desktop version and lists all wikis that are available. The star indicates whether the wiki is marked as favorite, whereas favorite wikis are always shown at the top of the list. An arrow next to the wiki indicates that this wiki contains subpages. After clicking on a wiki, the subpages are displayed in the navigation menu (cf. level 1 in the middle of Figure 4.22) and the categories of the main navigation are replaced.

At the top of the navigation menu the user can return to the previous level which is the main menu in this case. On lower levels the navigation always consists of the two categories current and subpages. The category current contains the selected wiki or wiki page. After clicking on this item the current wiki or wiki page is opened and the navigation menu disappears. The second new category called subpage lists all pages on the next lower level. Similarly, users can navigate to the next lower level in case the pages have further subpages. Subpages are always indicated with an arrow next to the title. Personalized information for the user are shown with the menu that can be accessed by clicking on the three dots in the navigation bar. This menu contains the alert with overdue tasks and the profile of the user which is shown in Figure 3.20.

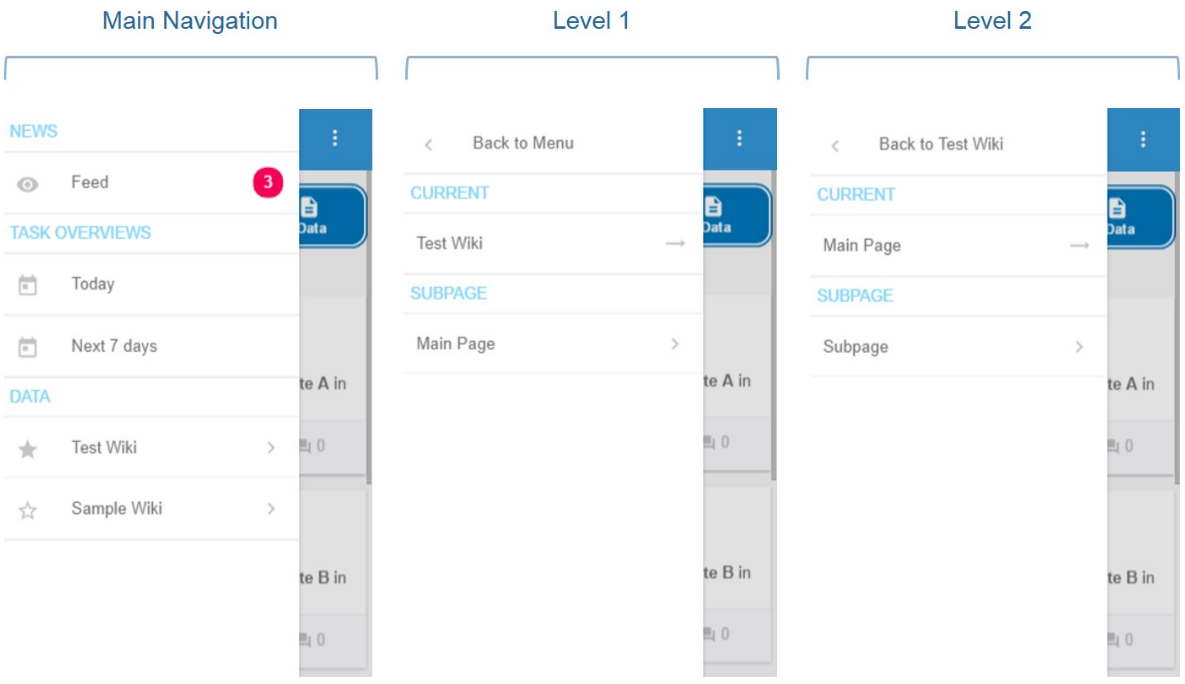


Figure 4.22.: Browsing to wiki pages on the lower level using the mobile navigation menu

4.3.3. Infinite Scrolling

Entries of the social feed in the mobile interface are dynamically reloaded, i.e., the user can scroll without recognizing the reloading of new entries from the server. Figure 4.23 illustrates the infinite scrolling with a sequence diagram. The *feedHtml* contains an AngularJS directive called *infiniteScroll* that displays the entries. This directive contains a function called *scroll* that is triggered when the user scrolls the feed on the mobile device. The scroll function contains a condition that compares the current position of the feed (*scrollTop*) with a variable *height* that contains the vertical height of the scroll bar. The condition evaluates to true after the current position is higher than the height divided by two, i.e., the user reached the middle of the scrollable height. In this case the directive invokes the method *loadMoreEntries* that is defined in the *feedCtrl*. The remaining operations of this method are not displayed in this sequence diagram since they are already shown in Figure 4.13. The method returns the entries from the server and feedHtml displays them in the browser. Therefore, user should not recognize that the entries are dynamically reloaded. The condition is tested with every scrolling that users performs in the social feed.

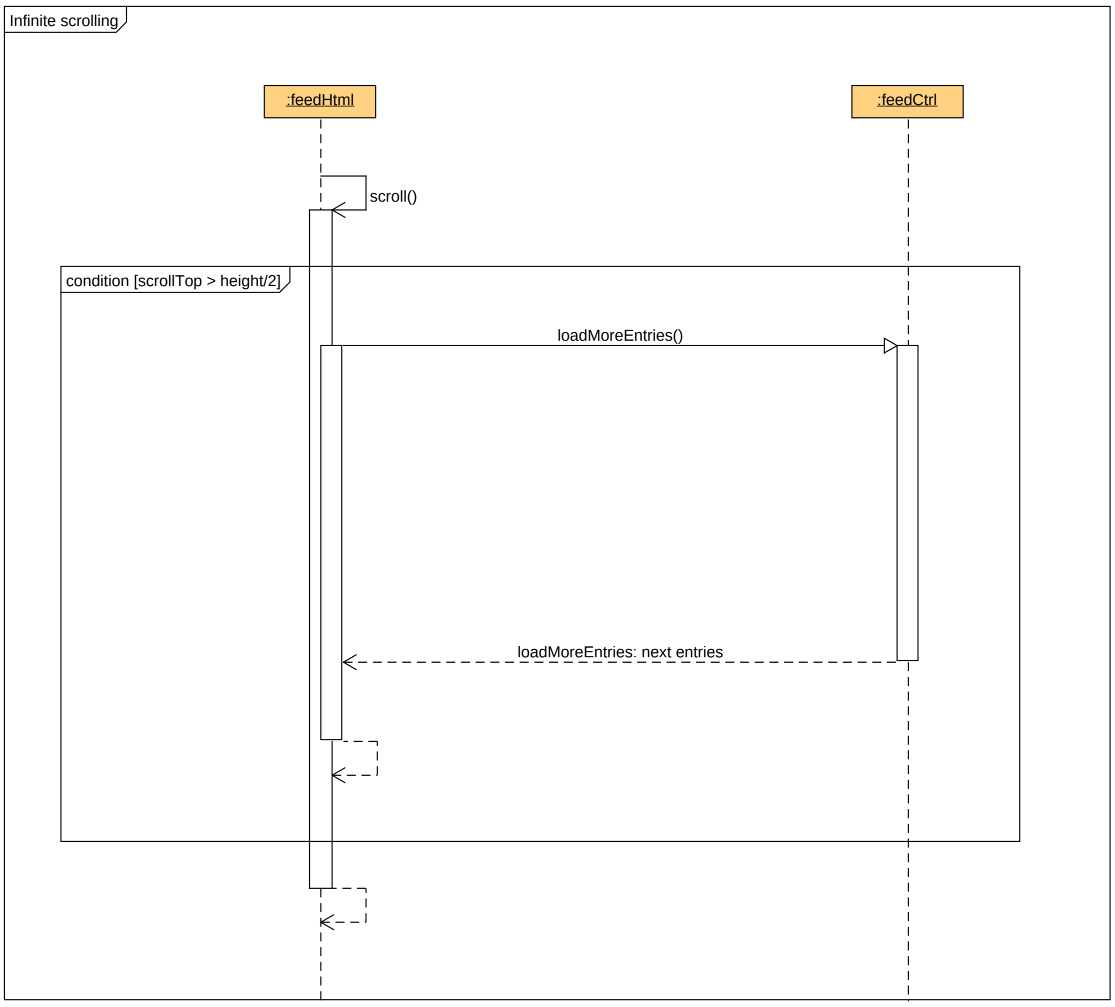

Infinite scrolling
:feedHtml
:feedCtrl
scroll()
condition [scrollTop > height/2]
loadMoreEntries()
loadMoreEntries: next entries

Figure 4.23.: Sequence diagram for the infinite scrolling of the feed in the mobile interface

A diverse system with multiple pathways and redundancies is more stable and less vulnerable to external shock than a uniform system with little diversity.

Donella H. Meadows

CHAPTER 5

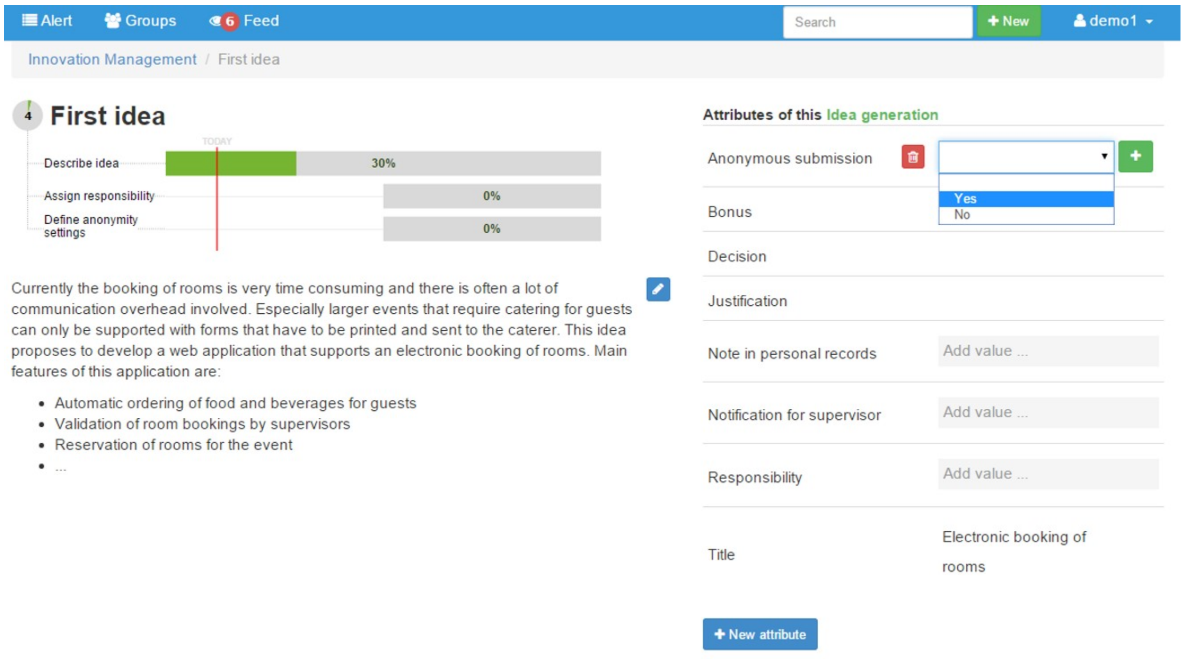
Case Studies

In this section three case studies are presented to demonstrate how the prototype is used in productive application scenarios. Goal of these case studies is to apply the prototype in existing KiPs to get some qualitative feedback. In the subsequent Section 6, the overall evaluation of the prototype is presented. Two of the scenarios are based on real world case studies that are conducted in two different organizations, while one of the scenarios uses an academic example. The description of the context and requirements for these case studies is explained in Section 2.2. In Section 5.1, the case study for innovation management is shown, which is based on the innovation process of a leading German software vendor. In this application scenario employees can submit ideas that are assessed by reviewers. The best ideas are selected and implemented in the organization. The second case study for EAM is explained in Section 5.2. The case study is based on a KiP within a German insurance organization. In this process a planned state of the entire EA is developed. The process is based on the Architecture Development Method (ADM) of TOGAF [Th09]. This ADM describes a process for enterprise architecture management that is used in many organizations. Section 5.3 presents the third case study that is based on artefact-oriented requirements engineering. In this process an artefact model for requirements engineering is provided, whereas the creation of these artefacts is coordinated with tasks.

5.1. Idea Generation

Within a master's thesis the innovation management process is investigated in detail [Ut14]. The innovation management process has three roles that are involved in the organization. The *idea applicant* has an idea that is submitted in the system and every employee in the organization is able to contribute his ideas. These ideas can be either incremental so that they are related to small improvements or disruptive. Disruptive ideas are concerned with innovations that have an impact on the business model of the enterprise. *Reviewers* are responsible for the assessment of ideas and they should be familiar with the subject of the idea. *Idea commissioners* determine suitable reviewers and make the overall decision whether the idea is accepted based on the reviews. Idea applicants might receive a bonus in case their idea is selected by the commissioners. Depending on the level of maturity of the idea some tasks might have to be skipped or added to the process. It is not possible that users are in more than one role for the same idea to avoid that users can review or choose their own idea. Figure 5.1 shows a screenshot of an idea after it is created in our software solution. Depending on the role of the logged in user, different tasks and attributes are visible in the work plan. The currently logged in user *demo1* is in the role of an idea applicant that wants to propose the *electronic booking of rooms* in the organization.

After the work plan is created the idea applicant has three tasks that have to be completed in the first stage. The task *describe idea* has a mandatory attribute *title* that has already been entered. In addition, the idea applicant can explain his idea more detailed in the text of the wiki page. Since this user already started with the description, the progress of this task is set manually to 30%. For this user the three attributes *bonus*, *decision* and *justification* are not editable on the work plan. The task *assign responsibility* has one mandatory attribute *responsibility* that has to be defined by the idea applicant. This is another person who is familiar with the area in which the idea is related to, e.g., a department manager. In the third task the *anonymity settings have to be defined* with three mandatory attributes for this idea. The *anonymous submission* is an enumeration that can have two predefined values that are shown in Figure 5.1. Anonymous submissions are necessary if idea applicants are afraid of consequences related to their submission. With the attribute *note in personal records* applicants can choose whether their proposal should be stored. Finally, applicants also have to define the attribute *notification to supervisor* according to their preferences.



First idea

Describe idea	30%
Assign responsibility	0%
Define anonymity settings	0%

Currently the booking of rooms is very time consuming and there is often a lot of communication overhead involved. Especially larger events that require catering for guests can only be supported with forms that have to be printed and sent to the caterer. This idea proposes to develop a web application that supports an electronic booking of rooms. Main features of this application are:

- Automatic ordering of food and beverages for guests
- Validation of room bookings by supervisors
- Reservation of rooms for the event
- ...

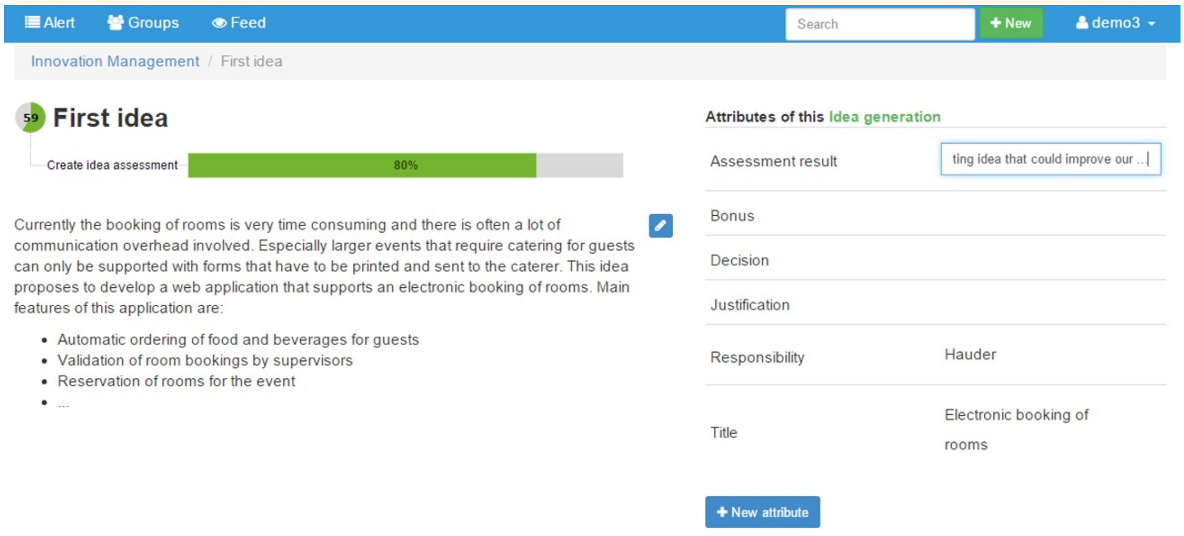
Attributes of this Idea generation

Anonymous submission	
Bonus	
Decision	
Justification	
Note in personal records	Add value ...
Notification for supervisor	Add value ...
Responsibility	Add value ...
Title	Electronic booking of rooms

Figure 5.1.: Creation of a new idea in the case study with three initial tasks

In case the idea has a low level of maturity the applicant can look for colleagues to further elaborate the idea. For this purpose the applicant can create new tasks in the work plan with the required expertise, e.g., for the creation of a business plan related to the idea. After the idea is described detailed enough the work plan continues with the reviewing of the idea. Initially the task *create idea assessment* is assigned to the idea commissioners that will delegate them in the most cases to a suitable reviewer. Usually, commissioners select reviewers based on the defined responsibility and the topic of the idea. Depending on the level of granularity of the idea, commissioners can estimate the duration of the task. In case more than one review is required, commissioners can create an arbitrary number of tasks and delegate them to other reviewers. This might be helpful if it is not possible to make the decision with the existing reviews or specific parts of the idea need to be evaluated by an expert. Figure 5.2 shows a screenshot of the work plan after the *create idea assessment* task is delegated to the user *demo3*. This user can see some of the attributes and the description of the idea on the wiki page provided by the applicant. It might be possible that the applicant creates additional attributes to upload documents, e.g., the business plan or screenshots of an early prototype.

Reviewers can update the progress of their task so that commissioners can monitor the state of the reviewing process. The assessment task has a mandatory attribute that contains the result of the review as simple string value, whereas the reviewer could also upload a file or enter his assessment on another wiki page. The task with the mandatory attribute is not visible on the work plan for the idea applicant. Nevertheless, applicants can see the progress of the idea that is shown in the small pie chart next to the title of the wiki page. The attributes for results of the assessment are not editable for reviewers, but they have read access to see the overall result after the entire evaluation process is finished. Depending on the idea some reviews might be more or less elaborated, i.e., some ideas can be rejected very quickly because they are not viable. Reviewers could create subpages with additional tasks to break down the required steps if the review is very extensive. In practice reviewers often involve experts in their teams during the assessment, which can be supported with subtasks.



First idea

Create idea assessment 80%

Currently the booking of rooms is very time consuming and there is often a lot of communication overhead involved. Especially larger events that require catering for guests can only be supported with forms that have to be printed and sent to the caterer. This idea proposes to develop a web application that supports an electronic booking of rooms. Main features of this application are:

- Automatic ordering of food and beverages for guests
- Validation of room bookings by supervisors
- Reservation of rooms for the event
- ...

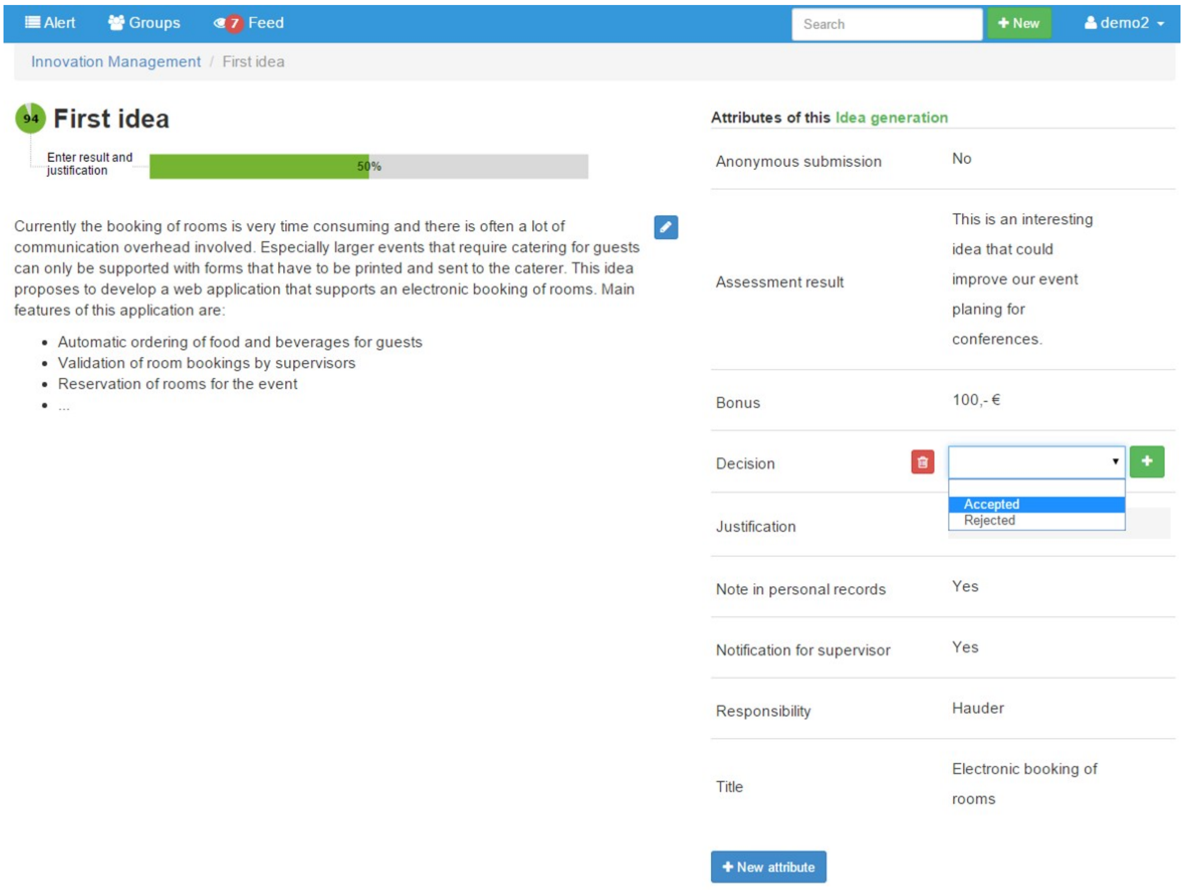
Attributes of this Idea generation

Assessment result	ting idea that could improve our ...
Bonus	
Decision	
Justification	
Responsibility	Hauder
Title	Electronic booking of rooms

Figure 5.2.: Reviewer creates an assessment of the idea in the mandatory attribute of the task

Figure 5.3 shows the work plan after the reviews are completed, whereas only one review is created in this example idea. The logged in user *demo2* is an idea commissioner that is responsible for the overall decisions. If the single review is not enough to make the decision, the commissioner might create new tasks for additional reviews if necessary. The task *enter result and justification* is linked with three mandatory attributes. In the attribute *decision* the commissioner can choose between *accepted* and *rejected*. Accepted ideas can be rewarded with a *bonus* if the idea has resulted in saving of expenses. In the organization 10% of the savings are usually rewarded with a bonus. The third mandatory attribute is the *justification* that needs to be provided for the decision.

In Figure 5.3 the progress for the idea generation is at 94%. This progress is visible to all stakeholders that are involved in the process. After the *enter result and justification* task is completed the user *demo1* can see the decision for his proposal. In the case study that we conduced for the innovation management process in [Ut14], this is the last task and the implementation phase for accepted ideas is not covered. In future work the prototype could be used for the implementation of the idea in the subsequent phases of the innovation process as well. The structure of the work template is already known to a large extent since the process is captured in two existing tools, whereas one tool is unstructured and the other tool has a rigid workflow. With this prototype the entire process could be captured within one solution.


Alert
Groups
7 Feed
Search
New
demo2
Innovation Management / First idea
94
First idea
Enter result and justification
50%
New attribute

Currently the booking of rooms is very time consuming and there is often a lot of communication overhead involved. Especially larger events that require catering for guests can only be supported with forms that have to be printed and sent to the caterer. This idea proposes to develop a web application that supports an electronic booking of rooms. Main features of this application are:

- Automatic ordering of food and beverages for guests
- Validation of room bookings by supervisors
- Reservation of rooms for the event
- ...

Attributes of this Idea generation

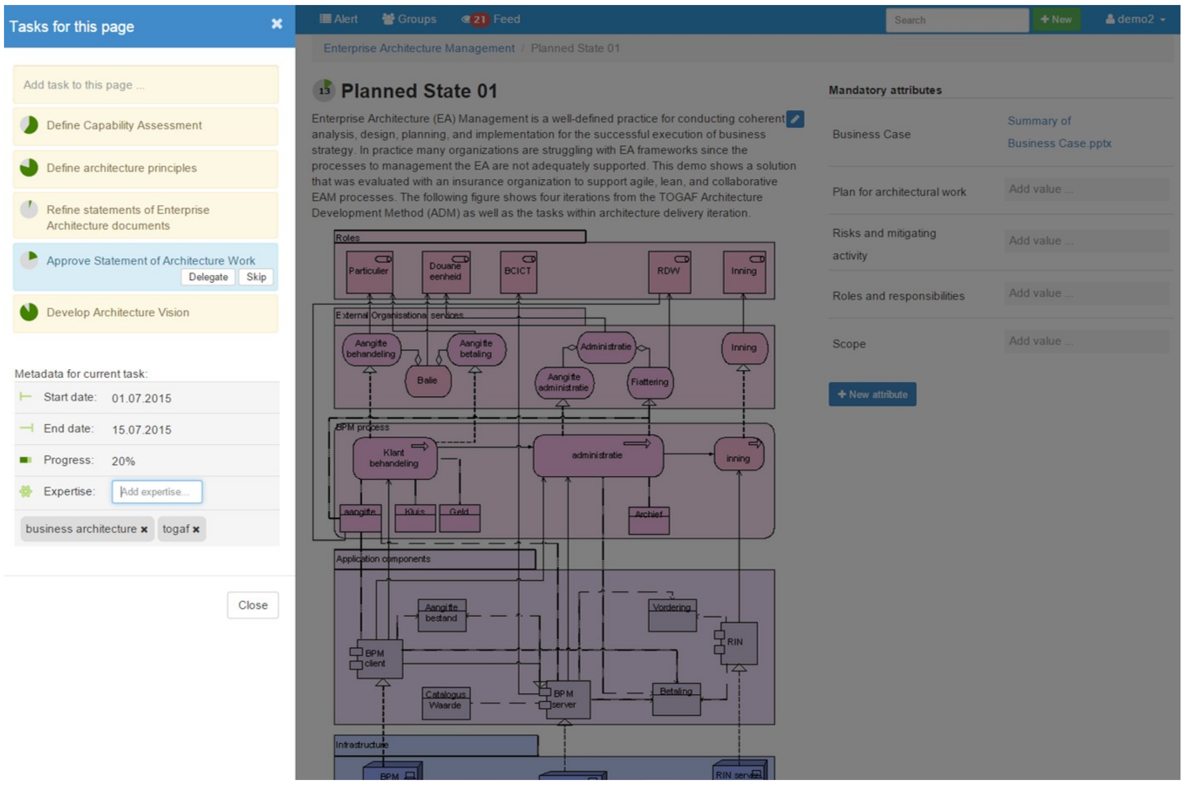
Anonymous submission	No
Assessment result	This is an interesting idea that could improve our event planing for conferences.
Bonus	100,- €
Decision	Accepted / Rejected
Justification	
Note in personal records	Yes
Notification for supervisor	Yes
Responsibility	Hauder
Title	Electronic booking of rooms

Figure 5.3.: Commissioner enters the decision based on the previously created review(s)

5.2. Development of a Planned State

A detailed description of the EAM use case with results of the evaluation from a German insurance organization can be found in [Ha14a]. The foundation of this process is the ADM that is described in [Th09]. The ADM consists of eight phases (A to H) that are iteratively executed. Before the process starts a preliminary phase called framework and principles is executed. The development of a planned state is mainly covered with the phases A to D. In phase A the *architecture vision* is developed or refined to get a common understanding about the goals of the initiative. In phase B the *business architecture* is developed with a target architecture and a roadmap that defines how this target state can be achieved. Phase C is concerned with the *information systems architecture* and derives requirements for the application architecture. Finally, phase D describes the development of the *technology architecture*. The subsequent phases E to H are not part of this case study since they are related to the implementation of a planned state.

Figure 5.4 shows a screenshot of the work plan for the development of a planned state that is initialized with some sample data. The work plan shown in this screenshot is currently in the architecture vision phase of the ADM. In this phase five tasks are enabled whereas every task has several artefacts attached that have to be created, e.g., the selected task *approve statement of architecture work*. This task has to create or refine five artefacts that must be uploaded as files. Another possibility would be to describe them on separate wiki pages. Since not all of tasks are necessary in the organization, it is possible to skip them on the work plan.



Add task to this page ...

Define Capability Assessment

Define architecture principles

Refine statements of Enterprise Architecture documents

Approve Statement of Architecture Work

Develop Architecture Vision

Metadata for current task:

Start date:	01.07.2015
End date:	15.07.2015
Progress:	20%
Expertise:	Add expertise...

business architecture ✖ togaf ✖

Enterprise Architecture Management / Planned State 01

Planned State 01

Enterprise Architecture (EA) Management is a well-defined practice for conducting coherent analysis, design, planning, and implementation for the successful execution of business strategy. In practice many organizations are struggling with EA frameworks since the processes to management the EA are not adequately supported. This demo shows a solution that was evaluated with an insurance organization to support agile, lean, and collaborative EAM processes. The following figure shows four iterations from the TOGAF Architecture Development Method (ADM) as well as the tasks within architecture delivery iteration.

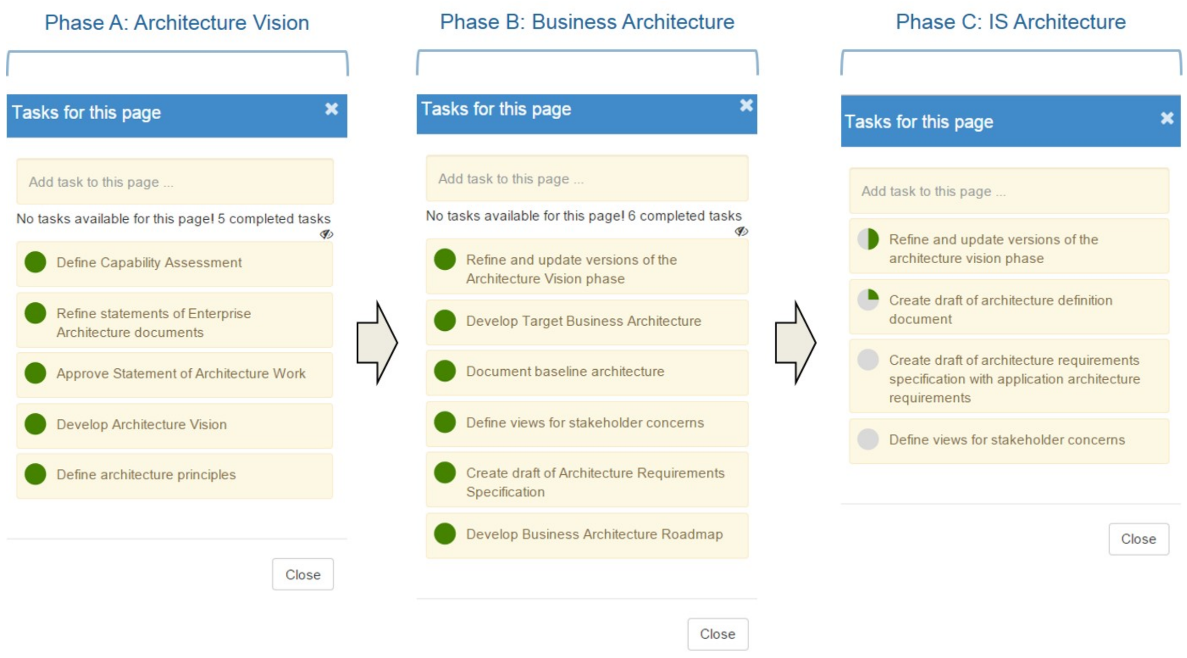
Mandatory attributes

Business Case	Summary of Business Case.pptx
Plan for architectural work	Add value ...
Risks and mitigating activity	Add value ...
Roles and responsibilities	Add value ...
Scope	Add value ...

Figure 5.4.: Architecture vision phase in TOGAF for the development of a planned state

The tasks for the phases in the ADM are structured with stages in the work template, i.e., every phase has its own stage that contains the tasks for this phase. Depending on the usage of the work plan in the organization, some tasks for a phase might be defined outside of the stage. This might be possible in case tasks do not have to be completed before the subsequent phase starts. Figure 5.5 shows an illustration how the tasks of subsequent phases are enabled while the work plan is executed. After all tasks of the phase architecture vision are finished (or skipped), the tasks for the subsequent phase business architecture are enabled. In this phase several tasks have to be performed according to the ADM. The task at the top called *refine and update versions of the architecture vision phase* has a dependency that is explicitly described in TOGAF [Th09]. The remaining tasks of this phase have no dependencies in this sample, whereas this could be changed if necessary. Similarly, tasks of the phase information systems architecture are enabled after all tasks in the preceding stage are completed (or skipped). In the same way tasks for the phase technology architecture are enabled.

On of the main challenges in EAM is that *"no formal steps exist for defining, maintaining and implementing EA and EA frameworks are not rigid enough in describing these steps"* [LKL10]. With the solution proposed in this thesis these formal steps for EAM can defined with work templates. Similar to the development of a planned state other EAM processes could be supported with Darwin. Due to the flexibility of work plans EAM processes can be easily adapted to the organizational context. Organizations setting up an EAM initiative can start with rather simple work templates. With increasing maturity of the EAM function work templates can become more elaborated. Since existing frameworks are not rigid enough in describing steps for the management of the EA, the approach presented in this thesis could be used to define executable EAM work plans. Frameworks like TOGAF can be used as blueprint for the initial creation of work templates and collaboratively refined to the context and goals of the organization.


Phase A: Architecture Vision

Tasks for this page

Add task to this page ...

No tasks available for this page! 5 completed tasks

- Define Capability Assessment
- Refine statements of Enterprise Architecture documents
- Approve Statement of Architecture Work
- Develop Architecture Vision
- Define architecture principles

Close

Phase B: Business Architecture

Tasks for this page

Add task to this page ...

No tasks available for this page! 6 completed tasks

- Refine and update versions of the Architecture Vision phase
- Develop Target Business Architecture
- Document baseline architecture
- Define views for stakeholder concerns
- Create draft of Architecture Requirements Specification
- Develop Business Architecture Roadmap

Close

Phase C: IS Architecture

Tasks for this page

Add task to this page ...

- Refine and update versions of the architecture vision phase
- Create draft of architecture definition document
- Create draft of architecture requirements specification with application architecture requirements
- Define views for stakeholder concerns

Close

Figure 5.5.: Tasks of the phases are enabled after their preceding phase is completed

5.3. Elicitation of Requirements

A detailed description of the case study for requirements engineering can be found in the master's thesis in [Bi14]. This case study is based on the artefact-oriented approach for requirements engineering presented in [Fe10]. In contrast to activity-centric requirements engineering, this approach has a stronger emphasize on the resulting artefacts of the requirements elicitation. The proposed approach has many similarities with data-centric processes introduced in Section 2.1.2. It contains tasks that are associated with artefacts to guide the creation of these artefacts. The authors provide reference models as blueprints for requirements engineering and guidance for the customization of these reference models to the project context on a conceptual level. Figure 5.6 illustrates how these reference models can be executed on a work plan. The reference model for artefact-oriented requirements engineering contains three artefacts for the *context specification*, *requirements specification* and *system specification*. Every artefact consists of a set of content items that have to be delivered during the requirements elicitation.

The metamodel for artefact orientation is mapped to our model as described in [Bi14]. The work plan for the context specification shown in Figure 5.6 consists of three active tasks. The artefact *stakeholder model* has already been created during the task *create stakeholder model*. This stakeholder model and further example models are available for download[1]. This blueprint can be used as starting point for a work plan that is tailored to the specific project.

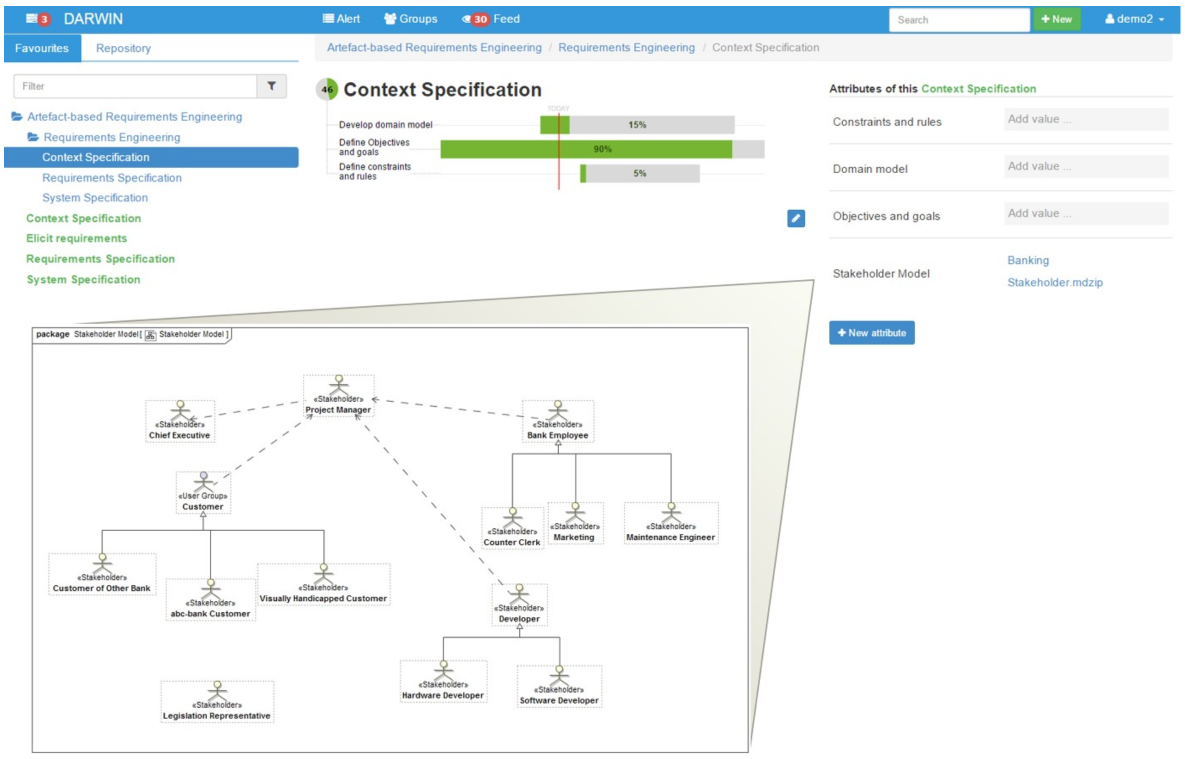

3 DARWIN
Alert
Groups
30 Feed
Search
+ New
demo2
Favourites
Repository
Artefact-based Requirements Engineering / Requirements Engineering / Context Specification
Filter
Artefact-based Requirements Engineering
Requirements Engineering
Context Specification
Requirements Specification
System Specification
Context Specification
Elicit requirements
Requirements Specification
System Specification
46 Context Specification
TODAY
Develop domain model 15%
Define Objectives and goals 90%
Define constraints and rules 5%
Attributes of this Context Specification
Constraints and rules
Add value ...
Domain model
Add value ...
Objectives and goals
Add value ...
Stakeholder Model
Banking
Stakeholder.mdzip
+ New attribute
package Stakeholder Model[Stakeholder Model]
«Stakeholder» Project Manager
«Stakeholder» Chief Executive
«Stakeholder» Bank Employee
«User Group» Customer
«Stakeholder» Counter Clerk
«Stakeholder» Marketing
«Stakeholder» Maintenance Engineer
«Stakeholder» Customer of Other Bank
«Stakeholder» abc-bank Customer
«Stakeholder» Visually Handicapped Customer
«Stakeholder» Developer
«Stakeholder» Hardware Developer
«Stakeholder» Software Developer
«Stakeholder» Legislation Representative

Figure 5.6.: Screenshot of the executable work plan for the context specification

[1]`http://www4.in.tum.de/~mendezfe/openspace.shtml`, last accessed on: 2015-07-06

Based on this generic approach for artefact orientation several instances of this model have been proposed, e.g., the business information systems analysis (BISA) [MF11b]. BISA is a reference model that proposes instances for artefacts that can be used in the context of business information systems. Figure 5.7 shows an excerpt of the BISA work templates in the prototype that are based on the proposed artefacts. The customized approach instantiates the BISA work templates for a specific project work plan. After the work plan is instantiated it can be dynamically tailored at runtime. Depending on the required level of detail they are two possibilities to model artefacts. In the first possibility artefacts are uploaded as entire file (cf. stakeholder model in Figure 5.6). In the second possibility artefacts are modeled with their work template that can be more structured (cf. business goal in Figure 5.7). In the latter case the degree of structure is higher since it is modeled on a separate wiki page and it can have its own attributes and tasks.

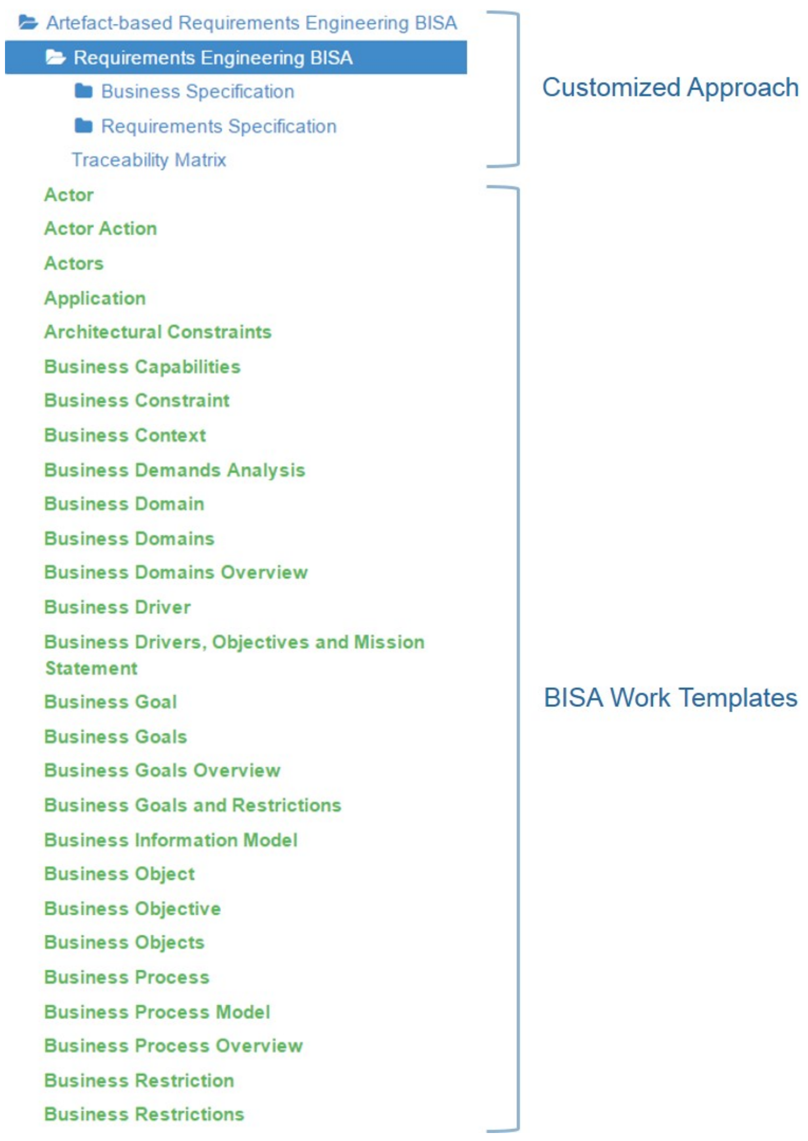

Artefact-based Requirements Engineering BISA
Requirements Engineering BISA
Business Specification
Requirements Specification
Traceability Matrix
Actor
Actor Action
Actors
Application
Architectural Constraints
Business Capabilities
Business Constraint
Business Context
Business Demands Analysis
Business Domain
Business Domains
Business Domains Overview
Business Driver
Business Drivers, Objectives and Mission Statement
Business Goal
Business Goals
Business Goals Overview
Business Goals and Restrictions
Business Information Model
Business Object
Business Objective
Business Objects
Business Process
Business Process Model
Business Process Overview
Business Restriction
Business Restrictions

Figure 5.7.: Excerpt of the available BISA work templates

To acquire knowledge, one must study; but to acquire wisdom, one must observe.

Marilyn vos Savant

CHAPTER 6

Evaluation

This chapter describes the evaluation of the software solution for the collaborative structuring of KiPs. While the case studies presented in the previous chapter seek to demonstrate the solution in potential application scenarios, the evaluation aims to measure the improvement for the support of KiPs with users in productive environments. The evaluation consists of four consecutive steps that are illustrated in Figure 6.1. In the first step the evaluation framework is developed and explained in Section 6.1. This framework contains the variables and values for the evaluation of the design science research artifact developed in this thesis. In the second step the preliminary study with seven participants is described in Section 6.2. The participants used the software solution productively over the period of approximately five months to develop a web application in a practical course at the Technical University Munich (TUM). The development process of the web application is highly unstructured without any predefined work template. In the third step the main study with 145 participants is described in Section 6.3. The duration and research variables of the main study are similar to the preliminary study, whereas the major difference is the larger scope of the main study. In the main study the software solution is used to support another practical course at TUM. The development process in the main study already consists of a predefined work template that contains mandatory tasks and deliverables. In both studies the participants are organized in teams and results are graded as part of their study program. In the final step a usability test of the mobile solution is performed and described in Section 6.4.

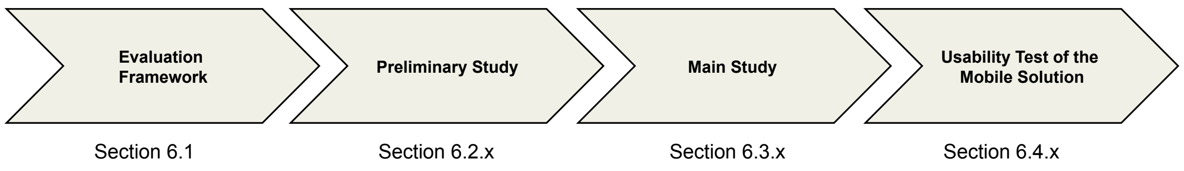


Figure 6.1.: Consecutive steps of the evaluation mapped to the sections of this chapter

6.1. Evaluation Framework

There are plenty of approaches available for the evaluation of design science research artifacts. In Figure 6.1, the evaluation framework used in this thesis is illustrated with the values that are applied in italic font which on [CGH09]. The approach can be considered as quantitative since the results are assessed through numerical values. The artifact focus is technical because the software solution developed in this thesis is evaluated. The artifact is instantiated to test the structuring concepts under real world conditions. The epistemological attitude is positivism, i.e., individual characteristics of evaluating persons are not considered. With the controlling function the evaluation examines certain criteria, e.g., flexibility, efficiency, information structure and work allocation. The method applied is the controlled experiment which we use to compare the performance of the software solution developed in this thesis with a control group. The evaluation object is the artifact itself, i.e., the software solution to structure KiPs. In the evaluation we are interested in a deployment perspective to analyze the usage of the artifact. The evaluation of the artifact is carried out externally by people that are not involved in the construction of the solution. Design science artifacts can be evaluated against three different reference points and in this thesis artifact against real world is chosen, i.e., to assess the suitability of the artifact in the real world. Finally, the time of the evaluation is performed ex post after the participants completed the experiment.

Variable	Value				
Approach	qualitative	*quantitative*			
Artifact Focus	*technical*	organizational	strategic		
Artifact Type	construct	model	method	*instantiation*	theory
Epistemology	*positivism*	interpretivism			
Function	knowledge function	*control function*	development function	legitimization function	
Method	action research	case study	field experiment	formal proofs	
	controlled experiment	prototype	survey		
Object	*artifact*	artifact construction			
Perspective	economic	*deployment*	engineering	epistemological	
Position	*externally*	internally			
Reference Point	artifact against research gap	*artifact against real world*	research gap against real world		
Time	ex ante	*ex post*			

Figure 6.2.: Evaluation framework for the artifact developed in this thesis [CGH09]

6.2. Preliminary Study

The preliminary study is based on the previously introduced evaluation framework from Section 6.1. We carefully followed the five steps for controlled software experiments as suggested in [Wo12], i.e., experiment scoping, experiment planning, experiment operation, analysis and presentation. The preliminary study is the first productive deployment of the software solution with end-users that were not involved in the construction of the system before. Results of this preliminary study were published initially in [HKM15]. The preliminary study has much less users involved than the subsequent main study, i.e., it is easier to comprehend how the participants are using the system to structure their processes. Another reason for conducting this preliminary study is the incorporation of feedback from initial users for the main study. In Section 6.2.1, the main objectives of the preliminary study are introduced. Section 6.2.2 covers the design of the preliminary study. The backgrounds of the participants in the preliminary study are explained in Section 6.2.3. Finally, the results and findings of the preliminary study are described in Section 6.2.4.

6.2.1. Objectives of the Preliminary Study

To the best of our knowledge there is no standardized procedure for the evaluation of systems that support KiPs available in literature or practice. A comparison with the requirements for software support of KiPs is also not expedient since the coverage of all requirements is out of scope for this thesis. The evaluation is focussed on the core research goal of this thesis, i.e., the collaborative structuring of KiPs through end-users. Therefore, we derived a set of questions from our main research hypothesis that are used to evaluate the following two objectives:

1. **Structure:** Structuring of KiPs is an important objective that we measure with seven variables in the preliminary study. With this structure work templates are generalized that allow to reproduce implicit knowledge of workers in an organization. Another important variable is the degree of information structure that can be defined in the solution. The structure of the KiP can also be used to visualize the progress of the process. Allocation of work is necessary to determine who should do which task based on certain criteria, e.g., skills or availability of users. The system should also provide guidance so that users are aware of what they have to do within the process. Finally, two variables are used to measure the efficiency and effectiveness of the solution.

2. **Flexibility:** Only considering the ability of a software solution to structure KiPs is not sufficient, since higher degrees of structure might impose worsening of the flexibility. For the measurement of the flexibility six variables are used in the preliminary study. Exception handling is necessary for knowledge workers to react on unpredictable situations. A variable for collaboration is used to measure the capability of the solution to support team work. Incorporating creative ideas within KiPs has to be possible in the solution. In agile environments knowledge workers have to be able to make incremental improvements. Another important requirement in agile environments is self-organization of teams, i.e., knowledge workers assign tasks collaboratively to each other without any central authority. Ease of use is a variable to measure how difficult the structuring of KiPs is perceived by the users.

6.2.2. Design of the Preliminary Study

The preliminary study has been conducted at the Technical University Munich (TUM) within two projects of the master lab course on web applications that takes place in every winter term. The projects started on 22$^{\text{nd}}$ October 2014 and were finished at the end of February 2015. Goal of these projects is the development of a web application using a scrum based process. Both teams have weekly sprint meetings with their advisor in which tasks are assigned to the students individually. During the period of the project 13 sprints have to be conducted by both teams. In addition to the development of the web application, both teams have to provide a written documentation of the project as well as an initial and final presentation. At the end of the project all of these deliverables are assessed by the advisors of the course for the final grading. One of the teams is using the solution Darwin that is developed in this thesis, while the other team serves as control group which is using a project management tool that is based on a kanban board. In the kanban board cards are used as tasks that are moved around the board. The objectives of the preliminary study introduced previously are evaluated for both projects separately. Thereby, the performance of Darwin as well as the research hypothesis can be tested through the comparison with the control group.

Both teams started with an entirely empty project respectively wiki at the beginning of the lab course, i.e., the students have to collaboratively structure this KiP in their teams. During the course of action the students created new tasks with their metadata and assigned attributes to these tasks on their own. Due to the low number of participants in the preliminary study, statistical conclusions are not possible in this study. Therefore, we conducted group interviews using open- and closed-ended questions with both teams separately after the projects are finished. Group interviews are selected to leverage the interaction of the project members in order to stimulate their experiences. Extreme answers are filtered out, because the participants have to agree on a common answer during the interview. During the project the students are carefully observed and their feedback is included early in Darwin. In addition, the structure of the work template for this KiP that emerged in the project within Darwin is retrieved from the system. The work template allows to assess whether it is possible for end-users without experience in business process modeling to structure a KiP.

6.2.3. Participants

The kanban as well as the Darwin team consist of four students that are all enrolled at TUM during the experiment. After about six weeks one of the members of the Darwin team dropped the course for personal reasons. The students are on the master level in computer science or information systems. The selection of the students is performed with a simple random sampling. None of the students has previous experience in business process modeling. The Darwin team was not involved in the construction of the system before the lab course started. Additionally, every team has an advisor from the chair of Software Engineering for Business Information Systems (sebis) at TUM. The advisor acts as product owner for the project and in the beginning the advisor is also responsible for the scrum master role. In later sprints this role rotates every week among the students and the advisor is only acting as product owner, i.e., every student becomes scrum muster at least once.

6.2.4. Results and Findings

The results for the variables related to the structure of the software solution are illustrated with the spider diagram in Figure 6.3. The dashed blue line indicates the results for the kanban board and the solid red line for Darwin. In Darwin the structuring of information is rated much better than in kanban since dedicated types can be created, e.g., presentations, sprints and mockups. The kanban team used a separate web storage solution to share files among team members in the project and copied links to the location of the files. In the kanban board solution the progress visualization is only supported on a coarse-grained level with columns, e.g., to do, doing and done. In Darwin the progress and the state of tasks is automatically updated based on the values of attributes. The allocation of worked is improved in Darwin through the usage of expertise tags that are assigned to tasks and the progress, whereas the kanban board only provides simple colors to indicate priorities. Guidance is also rated in favor of Darwin through the possibility to structure tasks in complex hierarchies with start and end dates. Efficiency is rated almost equally among both tools since the creation and the assignment of cards to different columns is very fast in the kanban board. A bigger difference can be observed for the variable effectiveness that is rated better for Darwin. According to the participants the reason for the good rating is the usage of mandatory results that are assigned to tasks. Reproducibility is rated better in Darwin through the reuse of sprints every week.

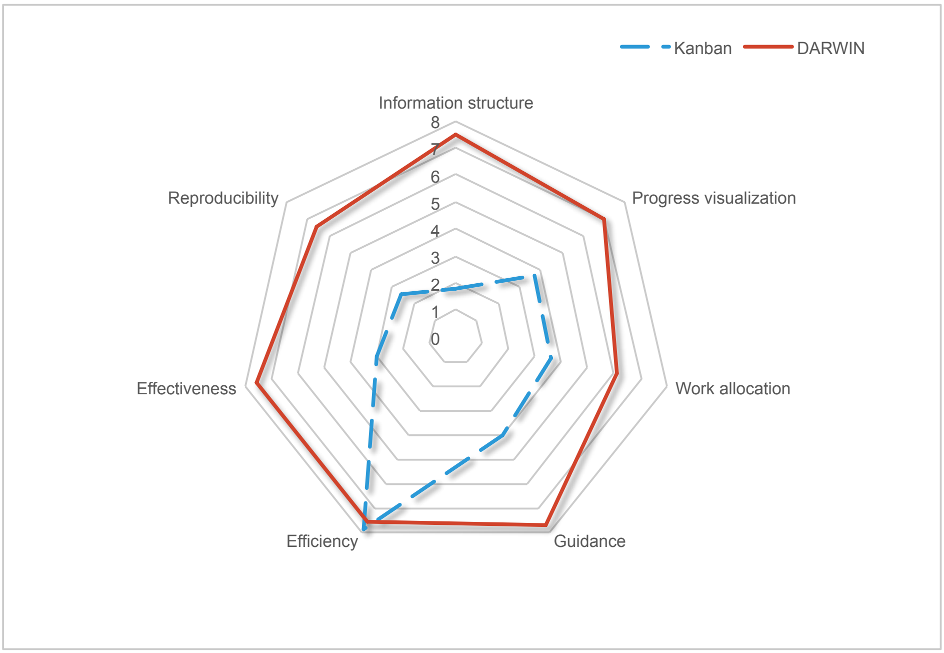


Figure 6.3.: Comparison of the structure for both solutions [HKM15]

To sum up, 6 out of 7 variables related to structure are rated much better in Darwin. Within 14 days a work template emerged with entities for, e.g., project, scrum and sprint. Examples for tasks that were created by students are "Add comment function to feed", "Refactor get-Expertises" and "Show completed tasks in profile". Attributes created by students are, e.g., "Mapping of features to design principles" (as file) and "(Project) acronym" (as string). More details regarding the emerged work template can be found in [HKM15].

Figure 6.4 illustrates the results for the six variables that are related to flexibility. The kanban board is rated slightly better in the handling of exceptions by the students. New cards can be easily created and moved between the columns to handle exceptions in the kanban board. Regarding collaboration there are no major differences that could be observed in the preliminary study. The solution with the kanban board also provides an activity feed with the latest actions. Creativity is rated in both solutions as equally well supported by the students. The rating for incremental improvements is slightly better in Darwin through the reuse and adaptation of sprints every week. Self-organization is rated almost equally for both solutions since all students created tasks on their own. Ease of use is rated very similar for both solutions again, whereas Darwin performs slightly better than the kanban board. According to the kanban team several functions are difficult to find in the solution. All of the 6 variables related to flexibility of the solution are very close to each other according to the students, i.e., Darwin provides a similar degree of flexibility compared to the kanban board.

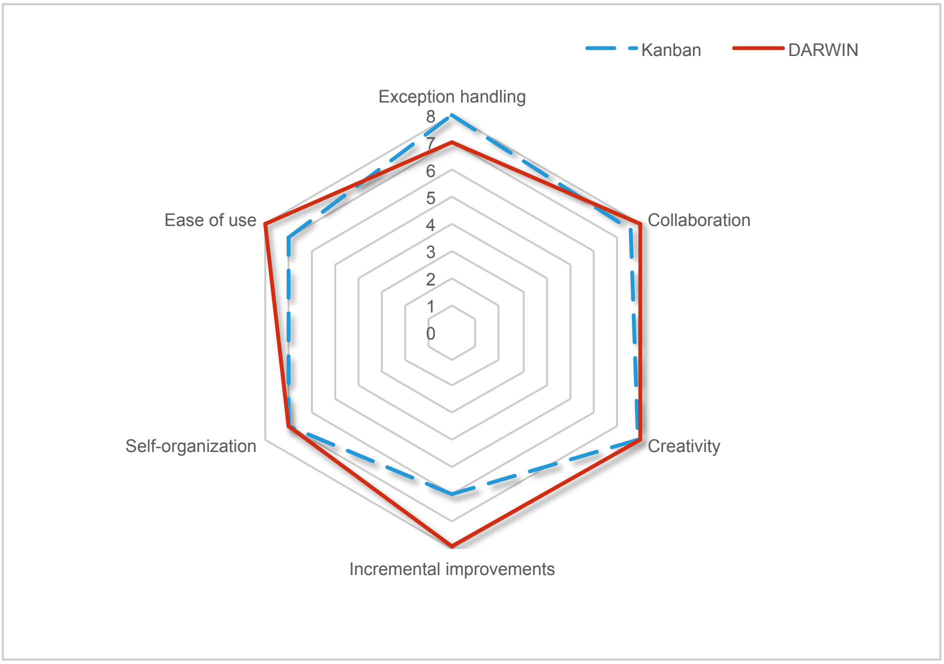


Figure 6.4.: Comparison of the flexibility for both solutions [HKM15]

The preliminary study confirmed our hypothesis that end-users without previous experience in business process modeling can be empowered to collaboratively structure KiPs. A suitable work template for this process emerged at the beginning of the project and is described in [HKM15]. The students rated the structure of Darwin unambiguously better compared to the kanban board. At the same time the flexibility has not declined according to the students, i.e., Darwin is better suitable to support this KiP. Due to the limited number of participants the results of this preliminary study cannot be generalized. In the scenario of the preliminary study the work template is entirely unstructured in the beginning. In future iterations of the master lab course the resulting work template of the preliminary study could be used as starting point for the projects. Within the main study of the evaluation the performance of Darwin with a predefined work template is tested, e.g., in order to investigate the impact on the flexibility.

6.3. Main Study

Similar to the preliminary study, the main study is based on the evaluation framework from Section 6.1 and we also followed the five steps for controlled software experiments [Wo12]. In contrast to the limited number of participants in the preliminary study, the main study is performed on a much larger basis in order to validate the promising initial results. Furthermore, the main study includes additional objectives that are not investigated in the preliminary study. In Section 6.3.1, the objectives of the main study are described. The design of the main study is explained in Section 6.3.2, followed by details about the participants in Section 6.3.3. The participants are allowed to use additional tools during the study and an overview about the usage of other tools is presented in Section 6.3.4. The presentation of results and findings is divided into two sections. Section 6.3.5 contains the results for the flexibility and Section 6.3.6 contains the results for the structure of the solution.

6.3.1. Objectives of the Main Study

The main study has three objectives that are derived from our main research hypothesis and the results of the preliminary study. Some aspects of the evaluation that are not covered in the preliminary study are included in the main study. Another difference to the preliminary study is the larger size of participants that allows to draw more reliable conclusions. The main study has the following three objectives:

1. **Usage of work templates:** In the preliminary study the participants started without any predefined work template and incrementally structured the process during the project on their own. With this objective the usage of work templates that are already predefined is investigated, i.e., are the users also able to use predefined work templates. This objective is measured by monitoring the usage of the system through Google Analytics[1] and documenting problems in the handling of predefined work templates.
2. **Structure and flexibility:** The objectives for the measurement of the structure and flexibility are repeated from the preliminary study in order to validate the performance of the system with a larger amount of participants as well as a different application scenario. The application scenario is different since the work templates are already predefined, i.e., the results for the objectives are probably different from the preliminary study. This is necessary since KiPs range from entirely unstructured to loosely structured in the process management spectrum (cf. Figure 2.2). Together with the preliminary study the evaluation of Darwin covers different degrees of structure for KiPs.
3. **Usability of the mobile solution:** Participants of the main study are primarily working with the desktop version of Darwin. For the evaluation of the mobile solution the usability is tested with a standardized procedure at the end of the main study. Results and feedback gathered through the usability test are used to improve the mobile solution in a second iteration. Furthermore, this usability test is a valuable opportunity to observe how the participants interact with the system through the application of the thinking aloud method.

[1]`http://www.google.com/analytics/`, last accessed on: 2015-08-07

6.3.2. Design of the Main Study

The main study is conducted within a lecture on web application engineering at TUM that takes place every summer term. This lecture consists of theoretical sessions and a practical project that covers all phases of a user-centered design process [DLH02]. The overall assessment of the students is only based on the outcome of this practical project, i.e., there is no written exam. The projects started on 13th April 2015 and were finished with the final presentation on 08th July 2015. The projects are structured through four exercise sheets that describe the required deliverables that have to be submitted. Table 6.1 briefly summarizes the content of the four exercises with the latest due date. Similar to the preliminary study all projects are divided into two separate partitions. The first partition is using Darwin for the submission of the deliverables, whereas the second partition acts as control group in the experiment and is submitting the deliverables through e-mail. For the Darwin partition a work template for the project with the four exercises is predefined for every team. Objective 1 and 3 are only evaluated with the participants of the first partition, while objective 2 is evaluated with both partitions independent from each other. Due to the high number of participants in the main study, group interviews were not feasible and we used an online questionnaire that is filled out by all participants of the lecture after the final presentation.

Name	Description	Due Date
Exercise 1	Development of a business idea by describing the value proposition and the customer segment. The business idea also contains four use cases that have to be implemented in the project. The results have to be summarized in a presentation and the contributions of each team member need to be documented.	27th April 2015
Exercise 2	The business model has to be described based on the Business Model Canvas (BMC). Four mockups have to be designed for the uses cases from the first exercise. A presentation is not required in this exercise, but the individual contributions of team members have to be documented.	13th May 2015
Exercise 3	A conceptual UML class diagram describing the information model needs to be created. One screenshot of a working use case has to be provided. Individual contributions of team members have to be documented and a presentation of the results created.	8th June 2015
Exercise 4	The final code of the web application has to be uploaded and screenshots of all use cases have to be created. A final presentation summarizing the results has to be prepared together with a live demo of the running web application. For all team members a documentation of the individual contributions needs to be created.	3rd July 2015

Table 6.1.: Description and due dates of the four exercises in the lecture for the main study

6.3.3. Participants

Initially the lecture started with 173 students that were registered and assigned to 44 teams. All of the students are on their master level and the majority has their field of study in computer science or information systems. Random sampling of the teams is not possible since the students are allowed to create own teams, e.g., with other students in the lecture that they already know. Teams 1 to 20 are assigned to the Darwin partition and the remaining teams submitted the deliverables by e-mail. Due to some students that dropped the lecture the main study was completed by 145 students. 75 of these students completed the main study with Darwin as member of one of the first 20 teams. Objective 1 and 3 are evaluated with these 75 students since they are familiar with Darwin. For the evaluation of objective 2 only participants that submitted at least 4 deliverables are selected to ensure high quality of the responses.

6.3.4. Usage of Other Tools

Figure 6.5 illustrates other tools that are used by individual students during the project from both partitions. The most frequently used other tools are WhatsApp[2] and Facebook[3] that were used for messaging. Followed by these tools are Dropbox[4] and Google Drive[5] that provide data storage services on the web.

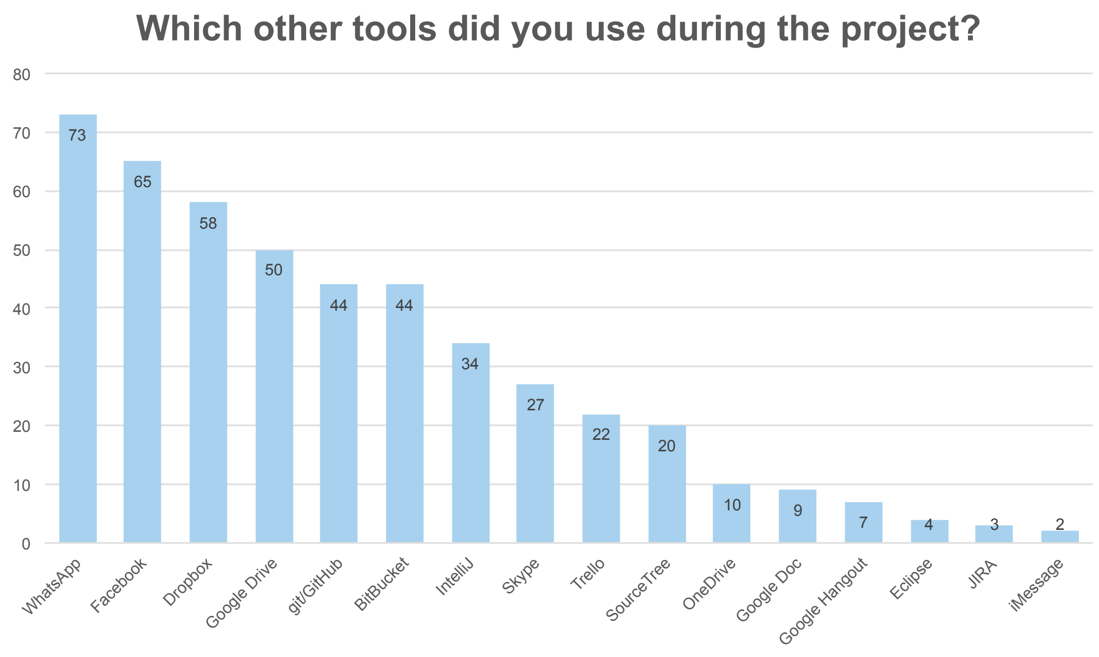


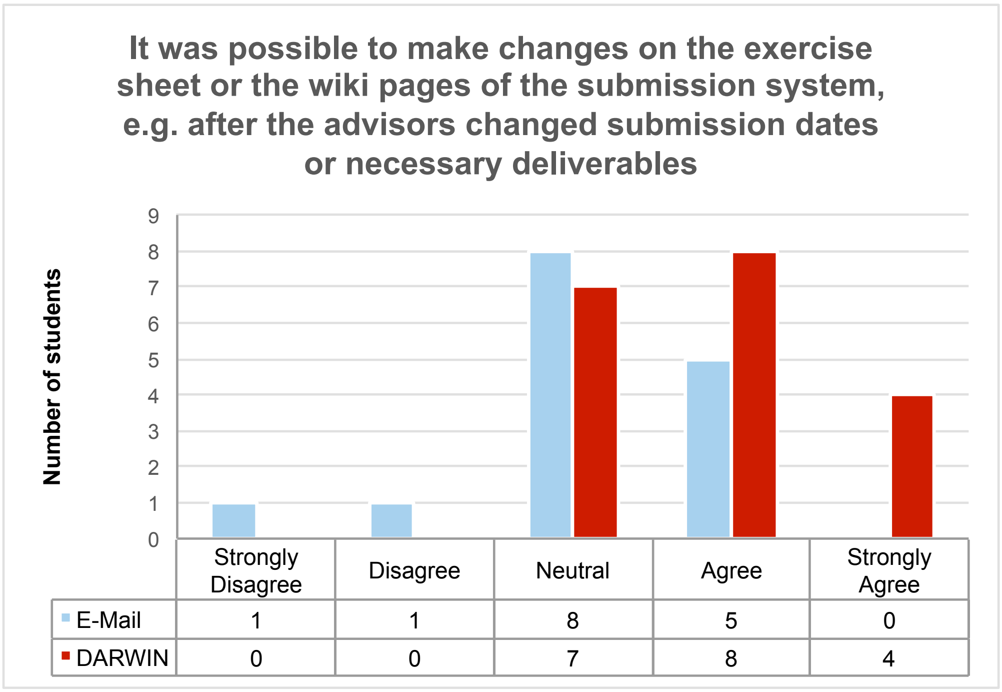
Figure 6.5.: Other tools that were used during the controlled experiments by all students

[2]https://web.whatsapp.com/, last accessed on: 2015-08-07
[3]https://www.facebook.com/, last accessed on: 2015-08-07
[4]https://www.dropbox.com/, last accessed on: 2015-08-07
[5]https://www.google.com/intl/de_de/drive/, last accessed on: 2015-08-07

6.3.5. Results and Findings for Flexibility

In the following the responses of the students for questions related to the flexibility are described. An overview about all results for this variable is illustrated at the end of this section with a spider diagram. Figure 6.6 illustrates the results for the handling of exceptions during the project, e.g., changes of the submission date or creation of new attributes for deliverables for the exercise sheet. On average Darwin achieves a better rating with 3.84 for this variable compared with 3.13 for e-mail. None of the students rated the exception handing in Darwin with disagree or strongly disagree. Several students mentioned that they would be able to perform changes if necessary. Examples for these changes that were performed are the creation of new deliverables on the exercise sheets or the update of the end date for tasks. In general, only few exceptions occurred during the exercises so that it was not necessary for the students to make changes frequently. This might explain the high number of students that responded to this question with neutral.

Another situation could be observed that indicates that the students are able to perform changes on the work plans for the exercises. Several students accidentally deleted attributes from the exercise sheet, because they wanted to remove the value from the attribute in order to upload a new version. All of the students were able to create the missing attribute again and assign it to the associated task. This is also an indicator that the deletion of the attribute is not intuitive enough since the delete button is to close to the attribute value. In future iterations of the lecture students could be encouraged to make more changes on the exercise sheets, e.g., to provide additional deliverables that are not explicitly mentioned by the advisors as necessary. In other application scenarios the handling of exceptions might be much more important, e.g., the scrum process in the preliminary study.

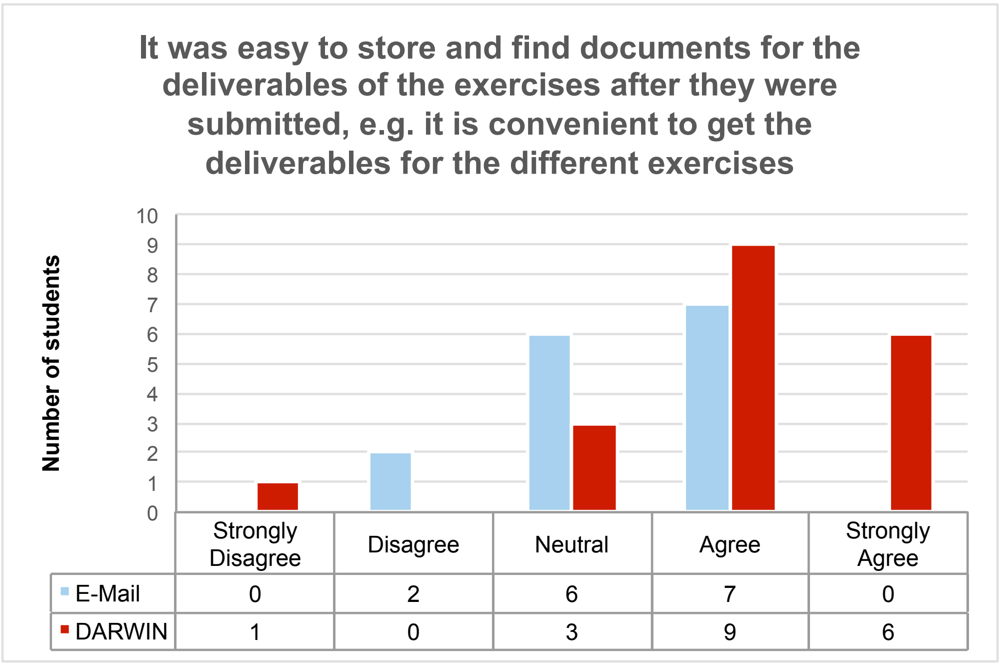


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
🟦 E-Mail	1	1	8	5	0
🟥 DARWIN	0	0	7	8	4

Figure 6.6.: Handling of exceptions during the project

Figure 6.7 summarizes the responses of the students for the information structure. Regarding the information structure Darwin is rated with 4 and e-mail with 3.3 on average. The majority of the Darwin users rated this question with agree or strongly agree. The reason for this good rating is the usage of attributes on wiki pages for the deliverables. All of the attributes for the students have the type file and the advisors of the course have additional integer attributes for the achieved points that are not visible to the students. Through the usage of access rights for attributes the members of a team are only able to see their own deliverables. Deliverables of other team members are hidden to students without access. The improvement of the information structure is also very valuable for the advisors of the lecture. The e-mail submissions are all sent to one mailbox that is prepared for the lecture and finding the right deliverables for the teams is quite inconvenient. Often teams submit several versions of the deliverables and this has to be taken into account by the advisors. In Darwin the latest version of the deliverables are attached to the exercise sheets and this saves a lot of time for the assessment of the deliverables. It could also be observed that many students uploaded new versions and deliverables very often downloaded by team members.

One limitation of the information structure in the current implementation that was mentioned by several students is the missing versioning function, i.e., it is not possible to revert deliverables to an earlier version. Deliverables that are deleted from the exercise sheet cannot be recovered again. This might be the reason why many teams only used Darwin for the final versions of the submissions and shared files through one of the other tools summarized in Section 6.3.4. All of the attributes for the students in the main study are structured as simple files. In future iterations the expressiveness of the data model provided by Darwin could be better exploited, e.g., through dedicated attribute types for the BMC or individual contributions of team members. This could limit the effort for students to generate files since the data could be directly added on the exercise pages or associated subpages.

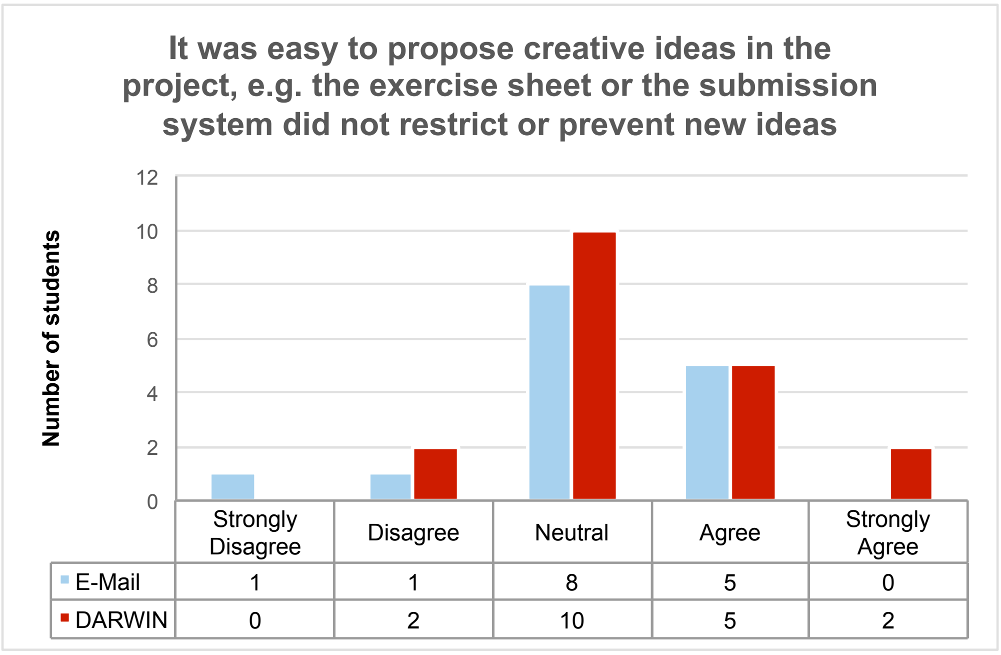


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
🟦 E-Mail	0	2	6	7	0
🟥 DARWIN	1	0	3	9	6

Figure 6.7.: Information structure for the deliverables of the project

The responses of the students related to creative ideas are illustrated in Figure 6.8. On average Darwin has a slightly better rating with 3.37 compared to 3.13 for e-mail. Only one student in the Darwin partition rated this question with disagree. Another student complaint that the exercise sheet has too many restrictions regarding the technology and more freedom would be desirable. The majority of students decided to respond with neutral, which might be an indicator that it was not necessary to incorporate creative ideas for the steps and deliverables of the exercises. In the preliminary study the incorporation of creative ideas in the projects is rated much higher for the scrum based development process. This might be due to the fact that the work template in the preliminary study was entirely unstructured at the beginning of the project. In the main study the work template is based on the experience of previous projects in the lecture that were organized in the same way, i.e., tasks and deliverables for the exercises are already practice proven and it is not necessary to incorporate any creative ideas to improve the work template. Therefore, the responses for this question are not unexpected since the objective of the main study is to investigate the usage of predefined work templates. All of the participants were able to execute the tasks on their work plans by uploading the required deliverables for the exercises.

A potential improvement for future iterations of the lecture is to support the early stages of the idea generation phase, i.e., the development and selection of the business idea that is implemented in the project. One possibility could be that every student needs to propose an own idea independent of the other team members. Similar to the case study introduced in Section 5.1, the proposed ideas could be reviewed with an innovation management process that is supported in Darwin. The other team members could act as reviewers and assess the ideas of their colleagues. The advisors of the lecture could act as idea commissioners and select the best ideas based on the reviews of the other students during the first exercise. After the selection of the best idea the team continues with this idea for the remaining three exercises.

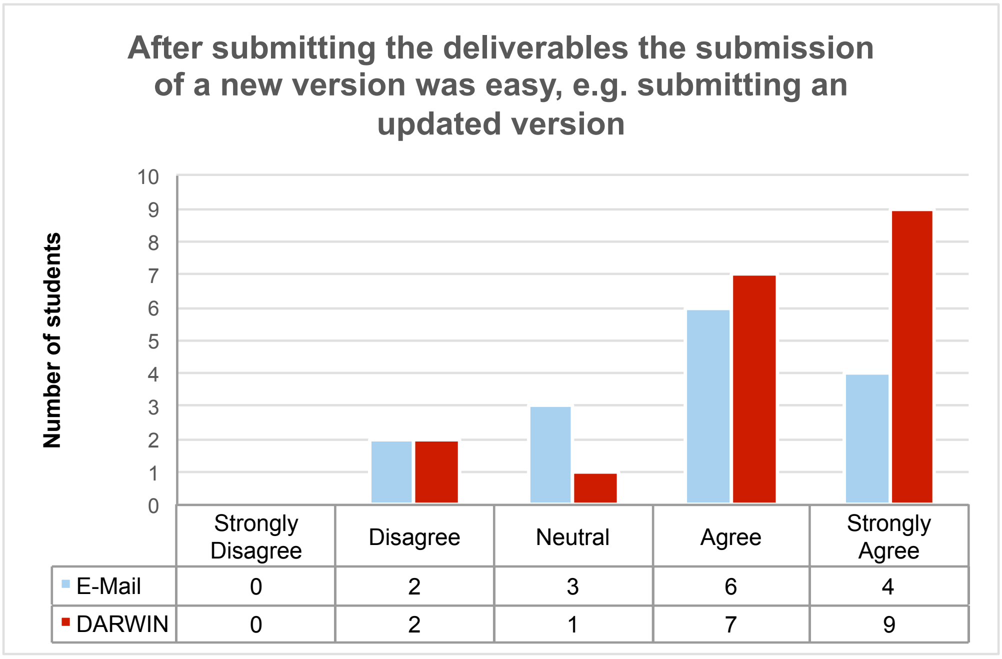


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
🟦 E-Mail	1	1	8	5	0
🟥 DARWIN	0	2	10	5	2

Figure 6.8.: Incorporating creative ideas in the project

Figure 6.9 illustrates the responses regarding incremental improvements of the submitted deliverables. Darwin achieves an average rating of 4.21 and e-mail 3.8 according to the students. The biggest difference among the responses can be observed for strongly agree, in which Darwin has more than twice as many ratings. During the exercises it could be observed that many teams submitted the deliverables more than once in Darwin, i.e., updated versions that are incrementally improved are submitted. Another reason might that the e-mail submissions are usually provided as a whole with all deliverables. In Darwin individual deliverables can be uploaded independently by the different team members of the project, e.g., some teams submitted the team sheet in the first exercise very early. Different members of the teams also submitted parts of all required deliverables. For the advisors the submission of new versions is much more comfortable in Darwin since the work plans always contain the latest versions of the deliverables. E-mail submissions of the teams always have to be checked for different versions to make sure that the latest version is assessed by the advisors.

Due to the splitting of the teams into two different partitions for the experiment, it was not possible provide detailed feedback for the early submissions of the deliverables. Otherwise the students working with Darwin would have an advantage that would be unfair for the final grading of all teams. In case all students in the lecture use Darwin for their projects in the future, the ability for the submission of incremental improvements could be used in conjunction with the social feed. Advisors of the lecture could provide feedback for the deliverables that are submitted early very efficiently. Every updated submission of a deliverable appears in the social feed and advisors can write feedback very quickly using the comment function. The realtime functionality of the comment function also allows discussions between students and the advisors in the context of the deliverables. Due to the high amount of students in the lecture, this would be much more difficult to achieve with communication that is based on e-mails.

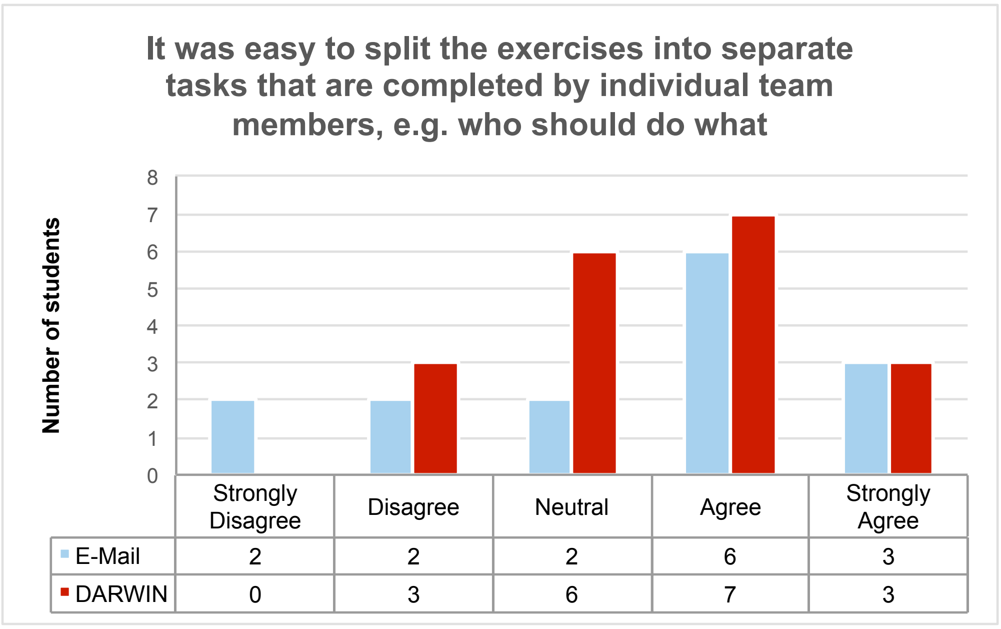


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
🟦 E-Mail	0	2	3	6	4
🟥 DARWIN	0	2	1	7	9

Figure 6.9.: Submission of incremental improvements for the deliverables

Results for the self-organization of the team are illustrated in Figure 6.10. With 3.53 Darwin achieves a slightly better rating than e-mail with 3.4 on average. Two students that are in the e-mail partition strongly disagree with this question. For these students it was very difficult to define explicit tasks for the team members. One of the reasons for the little improvement for this question could be instructions from advisors that are necessary to ensure that everybody contributes to the project. The advisors asked the participants that every student should work across all layers of the web application, e.g., it is not allowed that students only work on the front-end of the application. One student explicitly mentioned that this constraint makes it very difficult to split the exercises into separate tasks. Another reason for the little improvement could be the small size of the teams, e.g., some teams consist of less than four members in case a student dropped the lecture. Due to the small team sizes it might have been more productive to organize the tasks within face-to-face meetings, which would be much more difficult for teams that have more members.

Tasks on the exercises are assigned to the entire project team, i.e., every member of the project is in the execution role for the tasks. An improvement for future iterations could be the assignment of tasks in Darwin to individual team members using the delegate function. This delegation of tasks could be performed at the beginning of the exercises. Every exercise consists of one deliverable that describes the contributions of the team members. In case every student in the lecture uses Darwin in the future for the organization of the project, this deliverable can be replaced with the assigned tasks that are shown in the profile of the students. With this approach students would also encouraged to create additional tasks to coordinate the creation of deliverables. Due to the splitting of the teams into two partitions it was not possible to use this functionality since this might be unfair for the grading of the students, e.g., students not contributing to the project would be much easier to identify if they have no completed tasks in their profile.

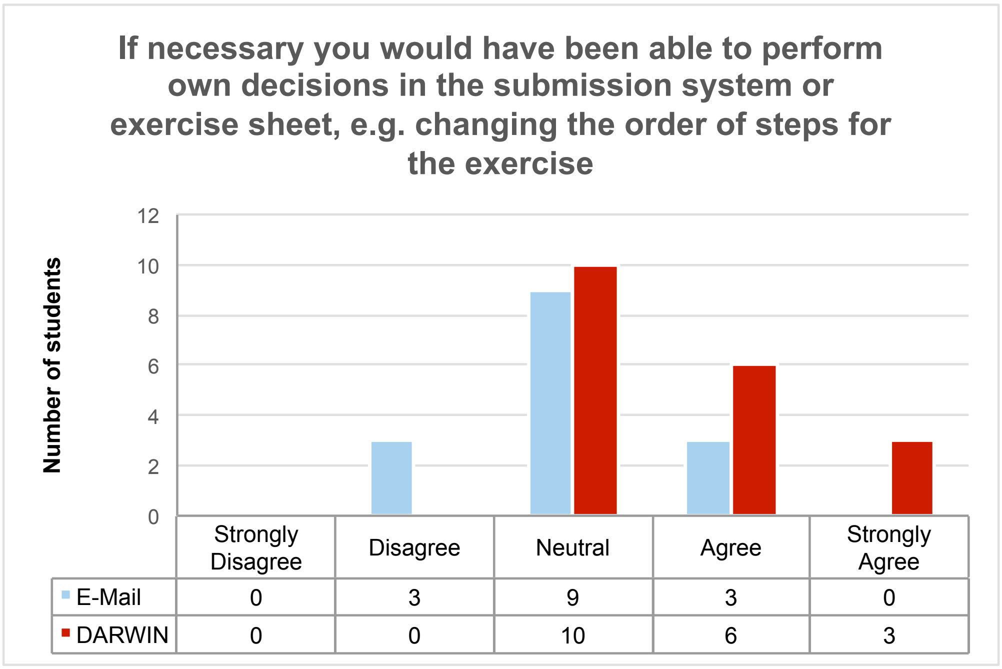


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
E-Mail	2	2	2	6	3
DARWIN	0	3	6	7	3

Figure 6.10.: Self-organization of the team during the project

The last variable that is used to measure the flexibility with the autonomy of the participants is illustrated Figure 6.11. Autonomous decision-making is an important issue for the support of knowledge workers. Darwin is rated with 3.63 and e-mail with 3 on average of the responses regarding this issue. 3 of the students in the e-mail partition disagree with the question and none of them strongly agrees, i.e., autonomous decision making is limited for the submission of deliverables by e-mail. None of the students working with Darwin responded with disagree or strongly disagree. 3 participants even strongly agree that they would have been able to perform own decisions in Darwin. Examples for own decisions are the order of steps in the exercise sheets or the creation of new tasks. A much higher value for this rating is not desired in this application scenario since the students have to complete the tasks by providing mandatory deliverables to pass the lecture, e.g., the skipping of tasks in the exercise is disabled for all students. Nevertheless, the order of steps is not enforced and students can change the attributes that are assigned to tasks. In other application scenarios a higher degree of autonomy can be enabled by allowing the skipping of tasks if this is desired, e.g., all tasks in the development process of preliminary study could be skipped by the students.

Darwin provides additional capabilities that allow to reduce the level of autonomous decision making that is desired for users. Less autonomy can be ensured by introducing logical dependencies between tasks through rules, e.g., the task "describe contributions of team members" has to be completed before the task "prepare final presentation" can be started. Another possibility is the clustering of tasks in stages that have dependencies, e.g., all tasks of an exercise have to be completed before the presentation for this exercise can be created. The usage of rules allows to ensure certain traces in KiPs without limiting the flexibility of the remaining tasks. The advisors of the lecture are tailors for the work template and they are responsible for the maintenance of the right degree of autonomy for students.



	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
E-Mail	0	3	9	3	0
DARWIN	0	0	10	6	3

Figure 6.11.: Making decisions autonomously within the project

All responses related to the flexibility are summarized with the spider diagram in Figure 6.12. The red solid line illustrates the results for Darwin and the blue dashed line for e-mail. All variables that are used to measure the flexibility in the main study could be improved through the usage of Darwin. In particular the autonomy, exception handling and information structure have the strongest improvement among all variables. We were able to observe that participants can use the predefined work templates and make adaptations on their own with very little training. The overall rating of the flexibility for the e-mail partition is 19.8, while Darwin achieves an overall rating of 22.6. This corresponds to an improvement for the flexibility of approximately 14.1% in this application scenario. The e-mail partition achieves a comparatively high rating for the flexibility which can be explained with several reasons.

The size of the teams is small in the main study and this makes the self-organization less challenging. The assignment of tasks on the exercise sheets to team members can be performed in face-to-face meetings. We expect that the improvement for this variable can be much higher for larger teams as soon as the face-to-face meetings are not feasible. Another reason is that the participants only used the system for the submission of the deliverables, e.g., tasks were not delegated to individual team members like in the preliminary study. In future iterations of the lecture every deliverable could be described with a separate wiki page. Team members that are responsible for the deliverable could create tasks on this separate wiki page to structure the process for the creation of the deliverable. Another advantage would be the transparency of the work distribution in the team, i.e., the actually completed tasks of every student are visible in their profiles.

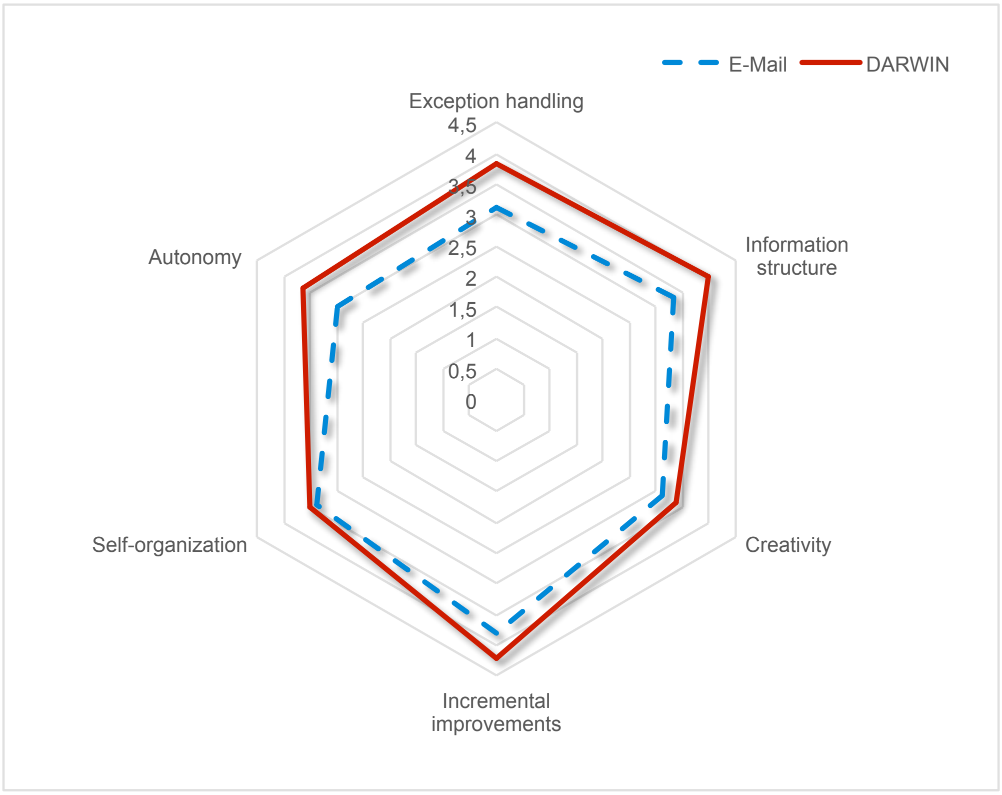
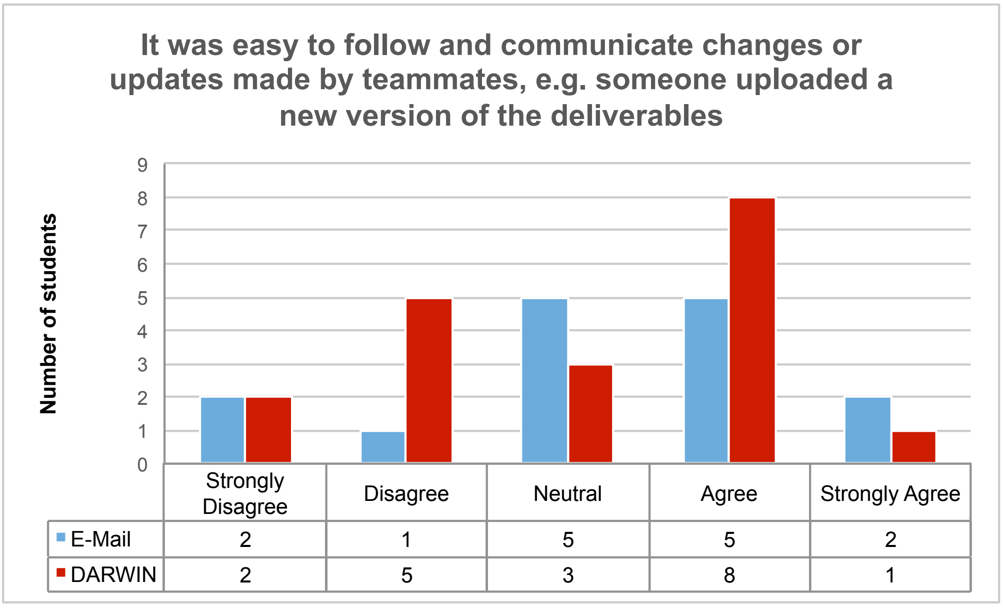


Figure 6.12.: Comparison of responses related to flexibility

6.3.6. Results and Findings for Structure

In the following the results for questions related to the structure of the solutions are described. Figure 6.13 illustrates the responses for the collaboration support during the project. Darwin is rated on average with 3.05 and e-mail with 3.27. Main reason for the worse rating is the disagree from several students that complained about too many notifications in the feed. This is a limitation of the current implementation, since the history events from every user in the system are shown. The high amount of notifications made it very difficult for students to follow and communicate changes made by their own teammates. Several students used the discussion feature of the social feed to raise questions about the exercises for the advisors. These questions were answered and visible to everybody in the system so that the amount of e-mail communication could be reduced efficiently. One of the students provided feedback through the discussion feature that a private discussion would also be very helpful for the internal team communication.

Based on the feedback of participants and questionnaire results regarding collaboration support during the project, future versions of Darwin need to include access rights in the generation and visualization of history events. In a first step data history entries should only be shown to users having read or edit rights for these entries. The same holds true for tasks history entries in the social feed, i.e., the user has to be within the execution role of the task. In addition, it might be helpful to introduce a watch feature so that users are able to explicitly decide which wiki pages they want to follow. Discussions in the social feed can be improved by providing the possibility to specify the visibility, e.g., which groups are able to read the discussion post. This would allow internal discussions with other members of the project team and public discussions that are relevant for all students. In case the number of history entries is still overwhelming for users, another possibility could be to determine which history entries are most relevant or interesting based on recommendation techniques.

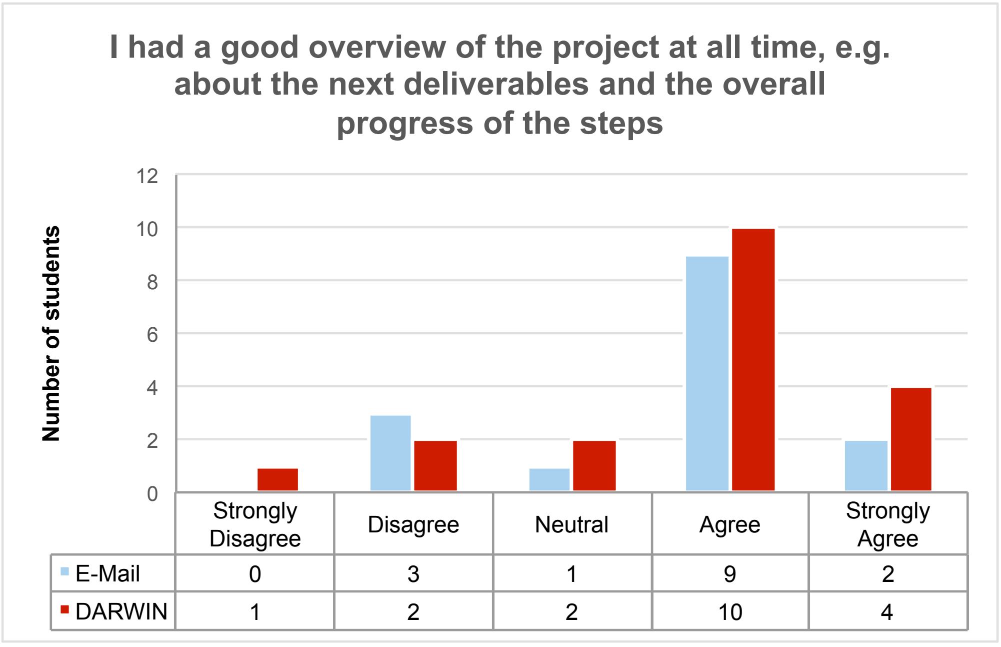


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
🟦 E-Mail	2	1	5	5	2
🟥 DARWIN	2	5	3	8	1

Figure 6.13.: Collaboration support during the project

Figure 6.14 illustrates the results for the visualization of the progress of the project. Darwin is rated with an average of 3.74 and e-mail with 3.67. In Darwin every project is described on a separate work plan with subtasks for the four exercises. In the timeline the start and end dates for every exercise are visible with the current progress. The current progress for the exercises and the entire project is visible with the pie charts on the wiki pages. During the lecture two questions of students occurred that are related to the same problem. After all tasks of an exercise are completed the progress of 100% is not reached. The reason for this is that every exercise work plan has an additional task for the advisors of the lecture. The task is necessary for the assessment of the deliverables and is not visible for students. These tasks are also considered for the overall progress of the exercise so that 100% progress are only reached after the advisors entered their assessments. Both students were not sure whether their exercises are completed after all tasks are finished because of the incomplete progress.

The first possibility to improve the progress visualization would be to compute the progress only based on tasks that are assigned to the currently logged in user. In our opinion this solution can only be provided in addition to the current progress visualization that includes tasks of other users as well, e.g., so that users can switch between personalized and cumulative progress visualizations. Only providing a personalized progress compassion might raise other difficulties for the users. In the second possibility remaining tasks of other users could be visualized in the timeline, e.g., by showing them greyed out in the timeline. Initially tasks of other users could be hidden and only indicated through an additional button with a counter. After clicking on this button remaining tasks of other users are visualized in the timeline. The overview about the project progress could also be improved if students could be encouraged to maintain the task progress manually since this feature was almost not used at all. For this purpose the static timeline could be extended with an interactive element that allows the maintenance of task metadata without the side window.

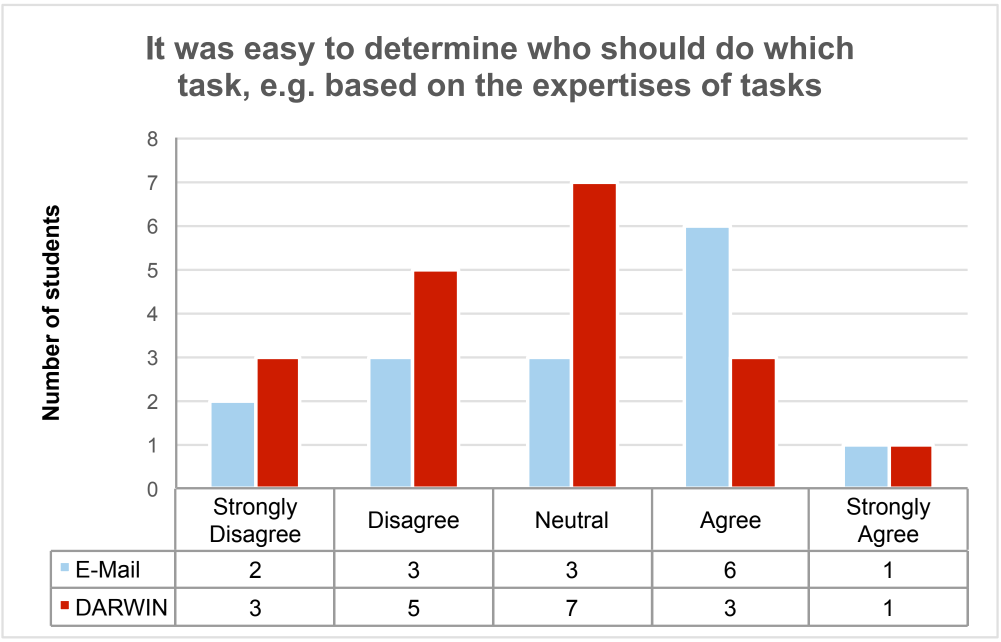


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
■ E-Mail	0	3	1	9	2
■ DARWIN	1	2	2	10	4

Figure 6.14.: Visualization of the progress of the project

Figure 6.15 illustrates the results for the allocation of work across team members of the project. Darwin is rated with an average score of 2.68 and e-mail with 3.07. Although every task in Darwin has predefined expertises that are required to complete the task, only one student strongly agrees with this question. Among all questions in the main study the average score for the allocation of work has the lowest score indicating that this needs to be improved in Darwin for the future. One student explicitly mentioned that the team worked on all tasks together so that they rated the question with neutral, i.e., it was not necessary to determine who should do which task. Another student strongly disagreed with this question for Darwin because they only used the system for the submission of the deliverables, i.e., students uploading the deliverables for a task might be different from students that actually contributed to the resolution of the task. This indicates that tasks in the work template are too coarse-grained and refinement with subtasks is necessary. Once more fine-grained tasks for the creation of deliverables are created in Darwin, the allocation of work during the exercises might be improved. This approach has already been very successful in the preliminary study but might require more training of the students.

In future iterations of the lecture deliverables should be created on separate wiki pages. Thereby, students can create subtasks for every deliverable and allocate tasks for the creation of the deliverables. With this approach the feedback of the student can be incorporated and skills of students in the profile would reflect the actual contribution of team members. In the preliminary study the usage of the scrum process positively influenced the allocation of work through the explicit planing phase in every sprint. The definition of knowledge work in Section 2.1.1 highlights the dual field of action, i.e., two separate phases called referential and actual field of action. While the actual field of action seems to be adequately supported, the referential field of action could be improved through an explicit planing phase for every deliverable in the project that is started on a regular basis or after all tasks are completed.

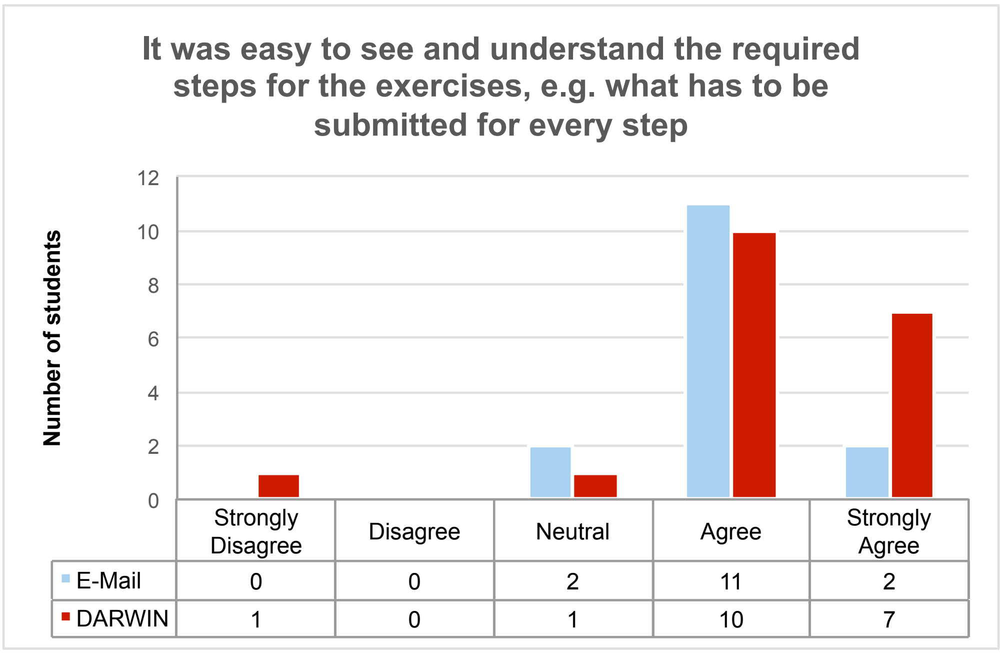


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
E-Mail	2	3	3	6	1
DARWIN	3	5	7	3	1

Figure 6.15.: Allocation of work for the team members of the project

Results for the guidance of the process during the main study are illustrated in Figure 6.16. Darwin is rated slightly better with 4.16 on average compared to e-mail with 4. The largest difference in the responses can be observed for strongly agree. Seven students gave the best rating for Darwin and only two for the submission with e-mail. The remaining choices are distributed very similar for this question, e.g., ten students agreed for Darwin and 11 for e-mail. Every step in the exercise sheets is represented as an own task in Darwin. Deliverables that have to be submitted by the students are assigned to tasks, i.e., after selecting a task in the exercise sheet the required deliverables are filtered. Tasks on the exercise sheets are only completed after all deliverables are uploaded on the wiki pages. The initial performance of the control group is already quite good since this application scenario is much more structured compared to the preliminary study in which the required steps are not predefined. None of the students in the Darwin partition submitted the deliverables wrong or incomplete, which is another indication that the guidance during the exercises seems to be successful.

Regarding the guidance during the projects Darwin provides a solid foundation for future improvements. In future iterations of the lecture deliverables can be structured more detailed on separate wiki pages, e.g., instead of a file the business model could be represented on an own work template that guides students in the creation of the business model. Another possibility without using separate wiki pages is the initialization with default values for deliverables. In this approach the attribute already contains a default file to describe the BMC, e.g., as word or editable pdf document. Students can download the file from the work plan and enter the empty fields. For the completion of the task this file can be uploaded again to replace the default file. Similarly, the exercise sheets contain detailed instructions how many slides are required for which step in the presentations. The structure of the presentation could be initialized with a default power point file that already contains the required empty slides. Both improvements could further advance the guidance during the process in the future.

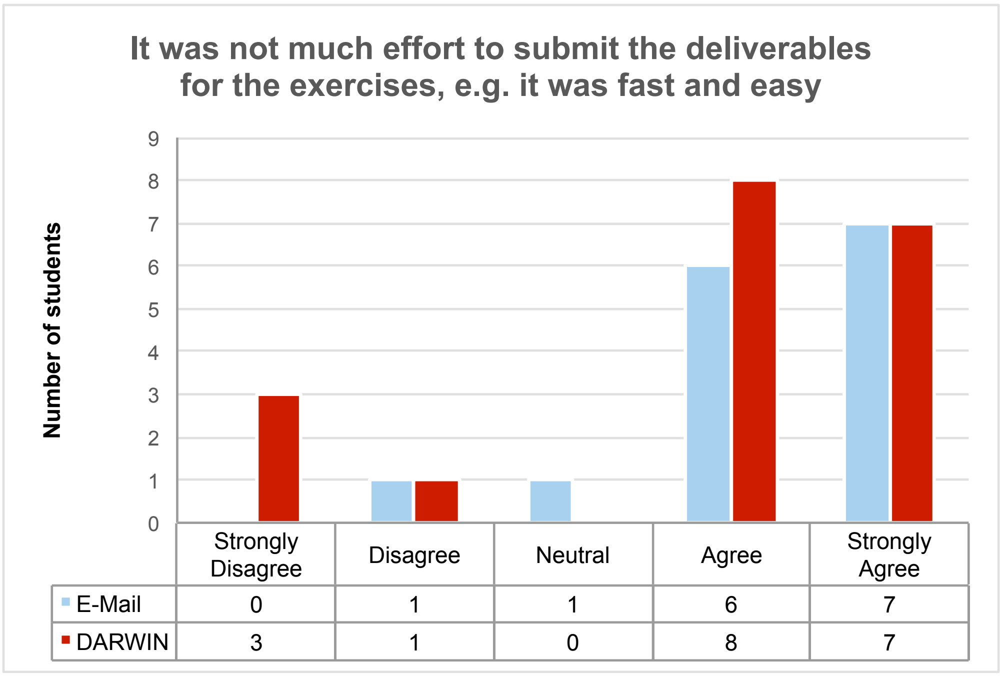


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
▪ E-Mail	0	0	2	11	2
▪ DARWIN	1	0	1	10	7

Figure 6.16.: Guidance of the process with the required steps in the project

Figure 6.17 illustrates the responses related to the efficiency of the solutions. Darwin is rated with 3.79 on average and e-mail with 4.27. Major difference between both partitions is that three students strongly disagreed for Darwin. Based on the feedback of the students we identified two reasons. The first reason is the missing progress bar for file uploads. Many of the deliverables are quite big, e.g., the code of the web application or screenshots for the implemented use cases of the solution. In the current implementation Darwin provides no feedback during the upload of the attribute. In some browsers the progress of the upload is indicated with a percentage at the bottom of the window, but this might have been overlooked by the these students. The second reason is that it was difficult for two students to figure out how the deliverables can be uploaded. Despite these problems the majority of users (15) agree or strongly agree that it was not much effort to submit the deliverables. The students in the control mentioned several times that the e-mail submission was fast and easy. Most of them are used to communication that is based on e-mail.

Although the majority of students confirmed that the submission of deliverables is not much effort in Darwin, four students were struggling with system. The first improvement that can be implemented quickly is related with the upload of files. Darwin needs to provide visual feedback about the progress of the upload. Furthermore, the entering of values for attributes has to be more intuitive for users that are not familiar with the system. In the current implementation empty attribute values are displayed with a gray box. Values of attributes that are associated to tasks could be highlighted on wiki pages, e.g., with red background color for the value. A more sophisticated solution would be the integration of Darwin with client applications, e.g., e-mail, word processing, drawing, or modeling tools. This would make the creation of tasks or upload of a new versions much more comfortable and faster. Disadvantage of this solution is the high effort for the integration with client applications.

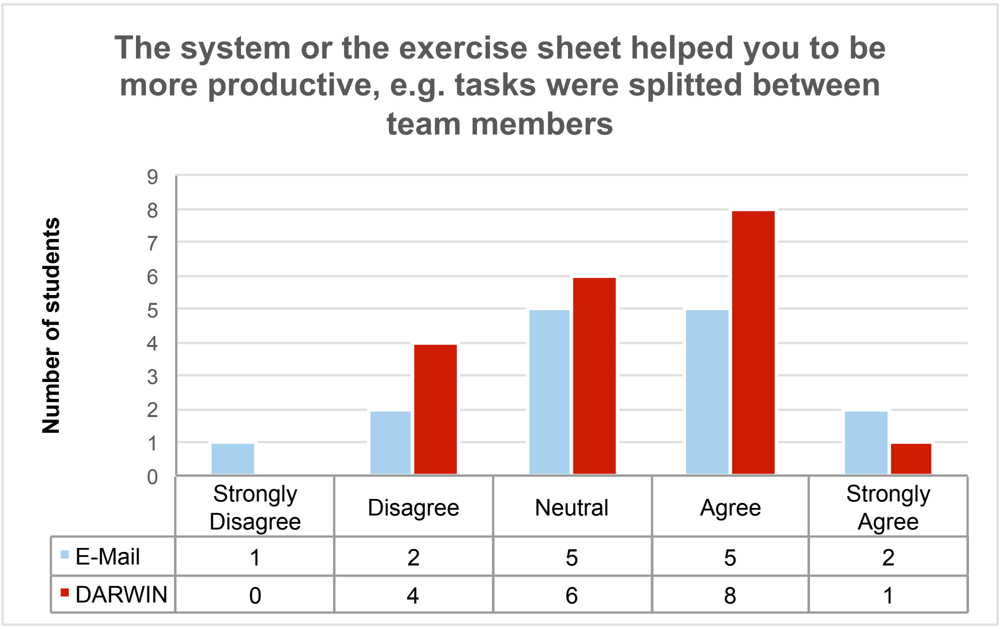


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
E-Mail	0	1	1	6	7
DARWIN	3	1	0	8	7

Figure 6.17.: Efficiency for the submission of the deliverables

Results for the effectivity of both partitions are summarized in Figure 6.18. Darwin is rated with 3.32 on average and e-mail with 3.33, so that both partitions are almost equally assessed by the participants. With eight students the majority of responses agreed that Darwin helped to be more productive for the submission of deliverables. Main goal of this question is to investigate whether the approach makes the students more productive in their projects. Several students mentioned that they worked on all tasks together and it was not necessary to split tasks. Due to several students that canceled the lecture for personal reasons that are not related to this evaluation, some teams had less than four team members at the end of the project. We assume that Darwin is better in improving the productivity for larger teams that consist of more members, because it becomes difficult to coordinate tasks within larger teams. During the project the advisors also asked the students to work across all layers of the web application to ensure that everybody participates in the project. To fulfill this requirement every student needs to work on at least one use case. These requirements limited the flexibility of the students and might have influenced the results for the productivity negatively.

In future iterations Darwin could help to improve the productivity with recommendations for tasks. These recommendations can be based on tasks that are created by other students in similar contexts. Recommended tasks can be shown in the worklist of every exercise sheet or the project. Users can select recommended tasks to see related metadata and attributes that are associated with the tasks. Initially, recommended tasks are not active in the work plan and users have two options for them. The first option is to delete the recommendations from the worklist in case they are not relevant. In the second option, recommendations that are useful for the work plan can be instantiated. The instantiation creates the task and associated mandatory attributes of the task. Based on the required expertises for the tasks, the system could automatically propose suitable team members that have the right skills. In addition to the expertises, start and end dates can be used to find team members with available resources.

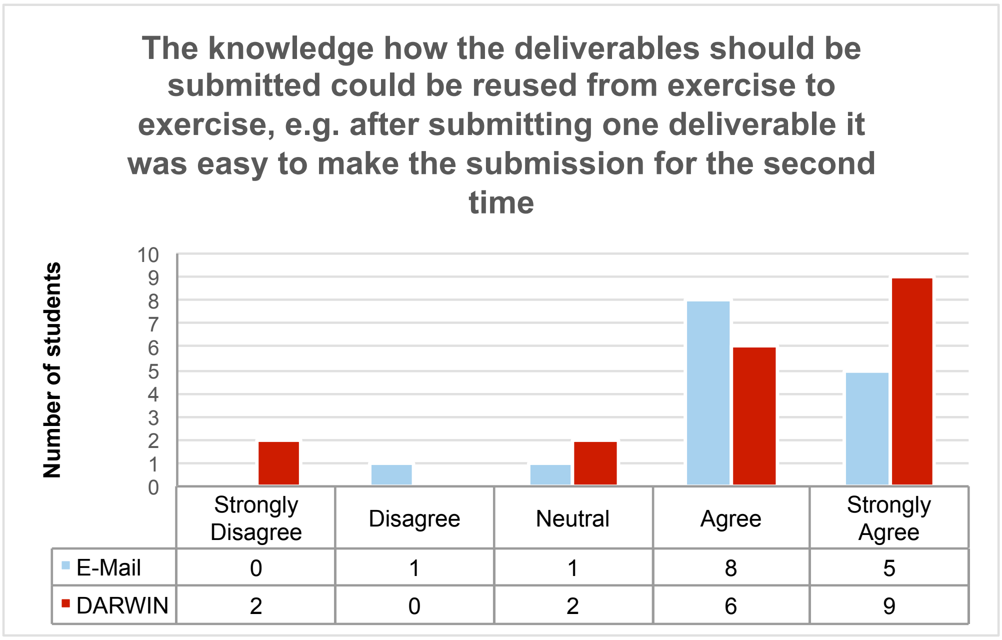


	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
E-Mail	1	2	5	5	2
DARWIN	0	4	6	8	1

Figure 6.18.: Effectivity of the steps for the submission of the deliverables

Figure 6.19 illustrates the results related to the reproducibility of process knowledge during the project. Darwin achieved an average rating of 4.05 and e-mail is slightly better with 4.13. The majority of users in Darwin agreed or strongly agreed with this question. One of the students explicitly mentioned that the submitting of deliverables was very intuitive. In the preliminary study the students reproduced sprints in the system, i.e., an initially created work template for sprints has been instantiated every week. Although the four exercise sheets have completely different goals, some aspects could be reused within the work template. Three of the four exercises required the creation of a presentation that summarizes the main results. In the second exercise a presentation was not required since there was no project meeting at the university. For every exercise sheet it was required to create a deliverable that describes contributions of team members, i.e., a dedicated type for contributions of team members can be reused in every exercise.

In future iterations of the lecture adaptations and improvements for work templates can be reused. These improvements can be based on changes that are made on the exercise sheets by students during the projects. The degree of structure for the work template of the lecture is currently on Stage III (cf. Section 3.1.2 on the evolution of KiPs). Stage IV can be reached by introducing rules for dependencies between tasks and more detailed constraints for attributes. Dependencies between tasks can be defined within the exercise, e.g., the presentation is always created after the other tasks are completed. A higher degree of structure can also contribute to an improved guidance for the lecture. Stage V introduces rigid types and strict attributes for work templates. Exercise sheets can be defined as rigid after they have reached a stable structure, i.e., students are not able to perform any adaptations or changes on the exercise sheets. For the creation of the deliverables on the exercise sheets Stage IV and V might not be desirable. These processes are highly unstructured and vary to a large extent for every project.



	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
🟦 E-Mail	0	1	1	8	5
🟥 DARWIN	2	0	2	6	9

Figure 6.19.: Reproducibility of process knowledge during the project

The results for all questions related to the structure are summarized in the spider diagram in Figure 6.20. The overall rating for Darwin is 24.8 and for e-mail 25.7. This corresponds to a small decline of 3.5% for the performance of the structure in the main study. The three variables collaboration, work allocation and efficiency have the biggest decline and need to be improved. The students provided valuable feedback for the improvement of these three variables in the future. All other variables for the structure are either better or equal, whereas they are considered as equal if the difference is smaller than 0.1. In contrast to the application scenario of the preliminary study, the structure with the tasks and deliverables for the exercises is entirely predefined. The exercise sheets contain the same information as the work template in Darwin. Therefore, we expected the much better performance of the structure for the control group in this application scenario.

With this result we can conclude that Darwin improved the flexibility without negative implications for the structure in the main study. Furthermore, the usage of predefined work templates with Darwin could be approved since all project teams were able to complete the tasks for the exercises. These positive results are a solid foundation for further improvements in the future. In addition to the submission of the deliverables, Darwin can provide support for the creation of the deliverables during the exercises in the next step. This can be accomplished by describing deliverables with separate work plans that can be structured by students, e.g., the development of the business model with the BMC. Another improvement is the integration of Darwin with client applications, e.g., e-mail and word processing tools. With this integration tasks can be created in the clients and deliverables automatically uploaded.

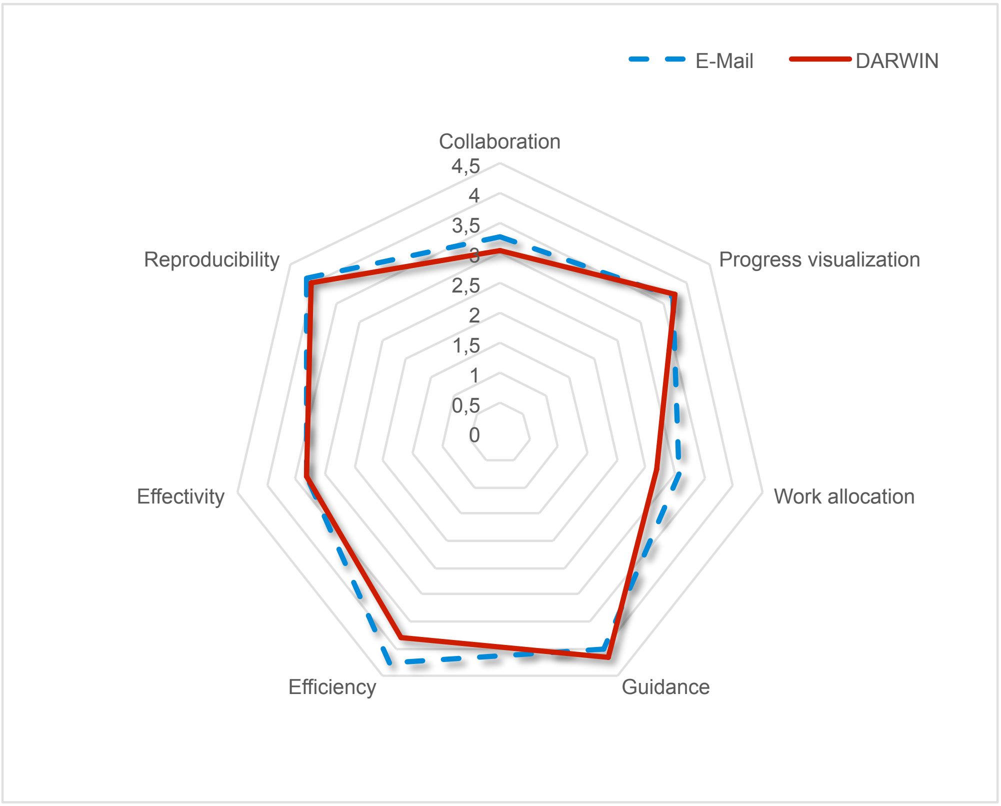


Figure 6.20.: Comparison of responses related to structure

6.4. Usability of the Mobile Solution

The final step of the evaluation conducted in this thesis is the usability test of the mobile solution. The development of the mobile solution started during the execution of the main study, e.g., it was not available for the students during the lecture. In Section 3.4.3, the user interface for the latest version of the mobile solution is introduced. This version already incorporates feedback gathered from the usability test, i.e., the test has been performed with an earlier version. The usability test has three main objectives that are investigated: First, the navigation structure that is designed for the mobile solution. Second, the creation of tasks with the maintenance of metadata for tasks. Third, the usage of the commenting function in the social feed. Before we conclude with the results and findings, Section 6.4.1 briefly describes foundations of usability testing that are relevant for this thesis. Our execution of the usability test with the tasks that have to be accomplished by the participants are explained in Section 6.4.2. Finally, the results and findings are summarized in Section 6.4.3.

6.4.1. Foundations of Usability Testing

Usability is an important characteristic for the quality of software products and according to the ISO 9126 standard it is *"a set of attributes that bear on the effort needed for use and on the individual assessment of such use, by a stated or implied set of users"* [IS01]. In this standard usability is divided into three attributes for understandability, learnability and operability. Understandability describes how much effort is necessary for users to comprehend and apply the logical concepts in the application. Learnability captures the amount of effort required for users to learn the application. Operability describes the effort of users to operate the software correctly. Another widely accepted set of attributes for the concept usability can be found in [Ni94]. In this publication usability consists of five attributes: learnability, efficiency, memorability, errors and satisfaction. Similar to the ISO 9126 standard efficiency describes the required learning time for new users. Efficiency measures the productivity gains that are facilitated by the interface. The extent to which users remember how an interface works is described with memorability. Errors are measured as unintended actions and satisfaction is used for the attitude of users regarding the interface.

Main goal of usability testing is the measurement of these characteristics and attributes. In this thesis the usability of the mobile solution is tested with questionnaires and the thinking aloud method. In the thinking aloud method users continuously speak about their impressions with the system, e.g., whether the meaning of colors is easy to understand. This makes it easier to find errors in the interface since it might be difficult for the participants to remember all details in a subsequent interview or questionnaire. Furthermore, we decided to use an online questionnaire since interviews would be too time consuming for this large number of participants. The questionnaire is based on the System Usability Scale (SUS), which allows reliable and low-cost assessments [Br96]. The SUS questionnaire consists of ten questions that are answered with a five point likert scale. Five of the questions are formulated positively and five negatively. The questions are adapted to the conditions of the specific test. The result of the questionnaire is a score between 0 and 100, whereas 100 is the best possible assessment. SUS scores between 60 and 80 can be classified from marginal to good. Scores above 80 indicate a very good or excellent usability of the software product.

6.4.2. Execution of the Test

The usability test was conducted on the 7th and 8th July 2015 in a controlled environment at TUM. 69 of the 75 students working with Darwin during their project completed the usability test, whereas six responses of the questionnaire are incomplete and removed from the dataset. The evaluation is divided into 12 slots that are executed on these two days. In every slot four to eight participants execute the usability test individually under the guidance of an instructed student. In the first step, participants receive several tasks that have to be completed in the mobile solution. Table 6.2 briefly describes these tasks with their test goal. Before the test started we deployed the latest version of Darwin that contains the mobile interface, i.e., students can log in with the same credentials that they use for their projects. None of the participants had previous access to the mobile solution or was involved in the design of the interface. We decided to test the mobile interface on desktop computers that are available in the controlled environment to avoid problems that might be related with specific smartphones. For the test we used the mobile device mode of the browser. Participants were asked to comment the usage of the system verbally for the thinking aloud method.

During the completion of the five tasks the instructed student carefully observed the interaction of the participants wit the system, e.g., to identify patterns of errors made by several users. After students completed the test using the mobile solution, they filled the online questionnaire that is based on the SUS in the second step. Based on the responses of the participants the SUS score is computed for the mobile interface. This score provides an initial indication of the usability performance that is a good starting point for more detailed tests in the future. Feedback that we gathered from the questionnaire and observations of the system usage are summarized in nine issues. Some of these issues are already addressed in the latest version that is presented in this thesis. All of the identified issues and their solution are described in Section 6.4.3.

Task	Instructions	Test Goal
Step 1	Please provide some short feedback by commenting the last discussion post of Matheus Hauder. The comments should be related to improvements for the lecture.	Find a specific discussion entry and write a comment
Step 2	Please open the wiki page that is used for the tasks and deliverables of exercise 4 in your project.	Navigation to the project and exercise page
Step 3	Create a task that called *Define test cases* with your name. Start date is today and end date is 15th July 2015. Required expertises are *Scala* and *unit testing*.	Create a new task
Step 4	Please use the task overview to check for any tasks that have to be finished until next week.	Finding the task overview for 7 days in the navigation
Step 5	Please finish the task called *Define test cases* that you have created previously in step 3.	Finishing a task by setting the progress to 100%

Table 6.2.: Test tasks that are performed by the participants in the mobile solution

6.4.3. Results and Findings

In this section results and findings for the usability of the mobile solution are summarized. The result of the SUS test is a score that provides an indication for the perceived usability of the system. We computed the overall score based on the responses of the 69 students in the questionnaire. The results of the test are related to an early version of the mobile interface, i.e., another iteration of the usability test with the latest version might reveal a better score. The mobile solution achieved an overall SUS score of 67.57 with a standard deviation of 10.36. This value is within the marginal and good area of the SUS range, i.e., this is an acceptable score but it leaves space for further improvements of the usability for the mobile solution.

We collected nine issues by observing the usage of the system during the test and gathered comments in the online questionnaire. Some of these issue are already resolved for the version presented in this thesis. The issues are grouped into five contexts in which they occurred and are described in the following:

1. **Login:** The sensitivity during the login was criticized since the mobile keyboard switches to capital letters at the beginning (issue 1). This is an issue since all login names provided to the students start with lower case letters. In the current implementation this issue is not resolved because the entire login needs to be much easier in the future, e.g., using existing accounts provided by other websites.

2. **Social feed:** Many students didn't use the filter functionality in the feed for the first task (issue 2). This was due to the fact that the filter buttons look like tabs and not buttons. In the current version this issue has already been resolved. The usage of the filter buttons was not consistent according to the students (issue 3). Sometimes clicking on a button enables filters and in other situations it disables them. This issue has been resolved by using the buttons consistently. Finally, the hiding of the like and comment button was not necessary since it requires an extra click (issue 4). Like and comment buttons are always shown to resolve this issue.

3. **Task overview:** In the overview tasks for today and the next seven days are listed. Initially tasks were not directly editable in the overview and users were redirected to the wiki page after clicking on a task (issue 5). This required an additional click in the user interface for the update of metadata. Direct editing of tasks is now possible in the overview and users are not redirected to the wiki page anymore.

4. **Wiki pages:** The tabs at the top of the wiki page were not immediately recognized by the students (issue 6). In the latest version the color of the tabs has been changed to highlight them on wiki pages. The closed tasks buttons were not highlighted enough on the wiki pages (issue 7). This issue has been resolved in the current version and completed tasks are much easier to recognize now.

5. **Creating and editing tasks:** The students expected that the editing of metadata is possible after the task is clicked, but clicking on a task only filters the mandatory attributes (issue 8). Editing the metadata requires an additional click and this could not be improved in the current version. Many students also struggled with the completion of tasks because they expected something like a "close task" button (issue 9). Instead of introducing such a button, we decided to work on the improvement of the progress functionality.

To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science.

Albert Einstein

CHAPTER 7

Critical Reflection and Future Research

In this chapter the thesis is concluded with a critical reflection and an outlook on future research. Section 7.1 provides the conclusion with a summary of all chapters and the final assessment of the seven research questions presented in the beginning. Possible limitations and threats to validity are provided in Section 7.2. The limitations are divided into three subsections for the conceptual part, the implementation and the evaluation of the approach. In Section 7.3, future research related to the results of this thesis is presented.

7.1. Conclusion

In Section 1, the motivation for this thesis is introduced with a detailed description of the problem, which is the collaborative structuring of KiPs through end-users. We applied the design science research methodology [Pe07] and explicitly stated seven research questions. Foundations for the thesis and related work are presented in Section 2. Fundamental topics for this thesis are KiPs, the integration of data and process as well as successful online communities. Requirements for the software support of KiPs are gathered from these three topics. The conceptual part for the structuring of KiPs is presented in Chapter 3. In the first step it describes the participating roles and the evolutionary approach for KiPs, i.e., the generalization of work templates through adaptable work plans. The required lightweight structuring concepts for this approach are presented in the second step, e.g., tasks, expertises, rules and stages. Based on social design principles from successful online communities, the features for the lightweight structuring concepts are described. The chapter continues with the design of the user interface including the mobile solution. Chapter 4 covers the implementation called Darwin by describing the most important aspects: the overall architecture, data model and CMMN workbench. Darwin is demonstrated in Chapter 5 with three case studies: idea generation, development of a planned state and elicitation of requirements. The evaluation of Darwin is described in Chapter 6 with the evaluation framework, preliminary and main study as well as the usability of the mobile solution. The participants in the preliminary and main study are divided into two groups that are compared against each other. In the preliminary study with 7 participants the structure could be improved considerably. In the main study with 145 participants the flexibility was improved by 14.1%. In the third step of the evaluation the mobile solution revealed an average SUS score of 67.57, which is in the marginal and good area of the test.

Research question 1 is answered with a set of 30 generic requirements for the software support of KiPs that are mostly derived from existing literature. The requirements provide a good foundation for the assessment of software solutions for KiPs. Based on three application scenarios that are investigated in detail, this set of requirements is compared against three application scenarios to answer research question 2. The relevance for the generic requirements differs for the application scenarios, i.e., not all requirements are equally relevant in these scenarios. An implementation of all requirements is not feasible within the timeframe of this thesis. One of the most challenging aspects for the software support of KiPs is the structuring at runtime through end-users, which is investigated with the remaining research questions. Research question 3 is answered with social design principles and patterns for successful online communities that are extracted from literature. These principles and patterns are used to design features in the user interface for the structuring of KiPs in order to motivate knowledge workers to contribute with their expertise and experience to the work plan. For research question 4, lightweight structuring concepts for KiPs are presented, e.g., tasks, expertises, attributes and roles. These concepts can be used by end-users to define and adapt work plans at runtime. Research question 5 is answered with an evolutionary approach for the structuring of KiPs in which work templates emerge from adaptations made by end-users on work plans. For this approach the participating roles for visitors, authors and tailors are extended from the Hybrid Wiki project. For research question 6, we developed an interactive CMMN workbench that can be used by modeling experts to maintain work templates. Research question 7 is answered with a mobile interface for Darwin that uses a subset of the available functionality.

In two studies Darwin as solution was applied with real world users in controlled environments and compared with existing solutions. The preliminary study was conducted with 7 participants and the main study with 145 participants. The already existing degree of structure is different in both studies. In the preliminary study the work template was entirely unstructured so that participants had to create own structuring concepts. The work template for the main study was already predefined and we investigated whether participants are able to use and adapt the work templates for their projects. In both studies one of the measurement variables could be improved without deteriorating the other one, e.g., the flexibility improved on the same level of structure in the main study. Therefore, our main research hypothesis for this thesis can be confirmed and we are able to show that end-users without knowledge about an existing process modeling notation can be empowered to collaboratively structure KiPs in controlled environments. These promising results can be applied in practice to improve existing generic case management solutions that require more flexibility at runtime. Tool vendors implementing the CMMN standard can incorporate the lightweight structuring concepts and the evolutionary approach proposed in this thesis. Another possibility is to apply the conceptual part of this thesis in vertical use cases, e.g., healthcare, software engineering, or idea generation. The proposed features for the lightweight structuring concepts can be incorporated in these systems to improve process support for specific application scenarios.

7.2. Limitations

In this section limitations and threats to the validity of the thesis are briefly discussed. The limitations are divided in three parts mirroring the chapters describing the core contributions of the thesis. Part 1 describes limitations that are related to the conceptual part. Limitations related to the implementation of the prototype are summarized in part 2. Finally, part 3 lists limitations related to the evaluation of this thesis.

1. *Structuring of knowledge-intensive processes:* The conceptual part of the solution is introduced in Chapter 3. It consists of the emergent structuring of KiPs, lightweight structuring concepts and features for the concepts that are embedded in the user interface. The emergent structuring of KiPs is based on the approach for Hybrid Wikis [Ne12]. Hybrid Wikis are used in production at sebis[1] since 2009 for the structuring of data models. In this thesis we enhanced this approach with structuring concepts for KiPs. The features incorporate widely accepted social design principles from successful online communities. The major limitation of this chapter is related to the formal description of the structuring concepts:

    a) The conceptual part lacks a formal specification of the lightweight structuring concepts, although the detailed description of the structuring concepts provides a solid foundation for the formalization in the future. With this formal specification process models can be checked for correctness, e.g., to avoid any inconsistent states in the work templates. Another advantage of the formal specification is the comparability with other related approaches. The Guard-Stage-Milestone (GSM) approach for the specification of business entity lifecycles provides a good starting point for the formalization. GSM is the foundation for CMMN and the structuring concepts provided in this thesis are a subset of this approach.

2. *Implementation:* The implementation of the conceptual approach is described in detail in Chapter 4. This chapter depicts the software solution with the core functionality, the workbench for CMMN and the implementation of the mobile user interface. An entire implementation of all requirements for the software support of KiPs exceeds the scope of this thesis. The main objective is the implementation of the conceptual part for the structuring of KiPs. The limitations of the implementation are described in the following:

    a) Sophisticated features for the management of work templates are not implemented in the current version. Although it is possible to define work templates in Darwin, there is no way to handle different versions or migrate already instantiated work templates. Depending on the application scenario, the duration of KiPs might be very long. In Darwin it is not possible to migrate changes of work templates on instances that are currently executed. This is a limitation that might prohibit the productive usage of Darwin for application scenarios with long duration. It is also not possible to manage different versions of work templates, which would be very helpful to analyze the evolution of templates.

    b) The second limitation related to the implementation is the functionality of the CMMN workbench. The workbench provides no automated layout of the shapes that are created so that it becomes confusing for modeling experts quickly. For pro-

[1]`https://wwwmatthes.in.tum.de`, last accessed on: 2015-08-17

ductive application of Darwin tailors require more detailed information about the usage of work plans, e.g., average completion time of tasks. In the current implementation the workbench is on the lowest conformance level of CMMN (cf. visual notation conformance in [Ob14]). As a consequence it is not possible to exchange work plans with cases in other tool solutions.

3. *Evaluation:* The results of the evaluation are summarized in Chapter 6. It starts with a description of the evaluation framework that is applied. In three serial steps the approach is evaluated with controlled experiments with real world users. A preliminary study is used to investigate a highly unstructured application scenario for a KiP. After feedback of the preliminary study is incorporated in the solution, the main study investigates an application scenario with a predefined work template. In the last step a usability test is conducted on the mobile interface of the solution. The following limitations related to the evaluation have to be mentioned.

   a) Throughout the evaluation we were not able to pass through all stages for the evolution of work templates, i.e., from Stage I to Stage V. This requires several iterations of the same KiP, which was not possible because of time constraints of this thesis. The preliminary as well as main study were only conducted once without any further iterations. The resulting work template that emerged during the preliminary study could be used for future iterations of the lab courses. This would correspond to an evolution from Stage I to III since an entirely unstructured process emerged to a work template with definitions. Another iteration of the main study could be used to show the evolution from Stage III to V. The work template for the exercises can be defined as rigid with strict attributes for the mandatory deliverables.

   b) Another limitation of the evaluation is the rather small team size in the preliminary and main study. None of the teams that used Darwin in practice had more than four team members. In practice many application scenarios of KiPs might have many more (or even less) knowledge workers that contribute to a common goal, e.g., scientific communities. A higher number of knowledge workers might hamper the allocation of work and the structuring of KiPs in case of missing common understanding. Another side effect of larger team sizes is that the structure of KiPs might become bigger, e.g., more tasks and attributes are created on deeply nested work plans. We believe that larger teams and projects can benefit more from the capabilities of Darwin, but this remains to be proven in practical settings.

   c) To the best of our knowledge there is no widely accepted approach for the evaluation and assessment of solutions supporting KiPs available. We decided to measure the performance of Darwin with two variables for the perceived structure and flexibility, since both are frequently mentioned in contemporary literature. There might be other relevant variables and measurement criteria that we didn't consider in our evaluation. Another limitation originates from the application scenario that we used for both studies. Although both application scenarios have different degrees of structure, there might be application scenarios for KiPs that perform differently for our approach. A common evaluation scenario for solutions supporting KiPs is highly desirable for the comparison of different approaches.

7.3. Future Research

The results and findings of this thesis provide manifold new opportunities for future work. In this section we focus on four possibilities for future research related to Darwin that are briefly described. On top of the four research areas several other topics are imaginable due to the high relevance of KiPs in practice and the limited amount of available research in this area.

7.3.1. Application in Practice

The first possibility for future research is the application of Darwin in practice. The evaluation in this thesis is performed with students in controlled environments at our university. An application in practice might reveal interesting findings that are not conceivable within a controlled environment. Furthermore, KiPs are omnipresent in organizations and many problems could be tackled with this approach. Starting points are already provided with the three case studies presented in this thesis in Chapter 5. In EAM existing patterns observed from practice are a good starting point for the definition of work templates. Darwin could be used as an execution platform for the EAM pattern catalog (EAMPC) presented in [Bu08]. The application of Darwin for the execution of the EAMPC has several advantages: the documentation of method patterns becomes much easier, since the work templates reflect the in reality performed processes in the organizations. Patterns can also be adapted to the context of organizations, which allows the creation of organization-specific methods, e.g., for specific industry sectors or concerns.

Another application in practice is the usage of Darwin for integrated care services (ICS). ICS can be supported with generic work templates that are adapted by professional case managers. The EU project NEXES[2] recently started an attempt to define the required logic models with BPMN [Ca15]. According to the authors this approach is not suitable for ICS due to the limited dynamic adaptation to changes that occur frequently for specific patients in integrated care. In future developments the authors attempt to follow an approach based on ACM that could be realized with Darwin. For the productive usage, Darwin needs to be connected with existing healthcare systems of hospitals to retrieve patient data, e.g., electronic health records (EHR). Clinical decision support systems (CDSS) have to be integrated in order to automatically analyze patient data. Results of these CDSS can be used to propose tasks in the individual treatment plans for patients. Patient data can also be gathered from remote monitoring devices in case of patients are not in the hospital. The integration with these external systems and devices can be realized with the API that is presented in Appendix A.

7.3.2. Recommendations

Main objective of Darwin is the structuring of KiPs at the hands of end-users. This structure can be exploited to provide recommendations for users in future research. Recommendations are beneficial for the evolution of KiPs since they contribute to the consistent usage of terms. For this purpose the similarity of users or work plans can be computed in a first step. In a second step the structure of related work plans can be recommended to suitable users. Rec-

[2] `http://www.webcitation.org/6O412APYq`, last accessed on: 2015-08-18

ommendations can be used for attributes and tasks on wiki pages, i.e., recommended tasks are shown with dashed lines in worklists. Users could instantiate recommended tasks and attributes by simple clicking on them in user interfaces. Metadata for tasks could be recommended with frequently used expertises or estimated durations of previously completed tasks. Similarly, expertises of tasks can be used to recommend suitable users based on their skills in the profiles. Initial work in this area has already been published for collaborative processes in IT service management [MNB11]. In this paper next best steps and expert recommendations are used to share knowledge about the resolution of cases. The recommendations are computed through annotated step flow models that are used to find matching paths. Another recent publication investigates recommendations in the context of ACM [HFH15]. In this approach log files are analyzed for the recommendation of individual steps. A major limitation of these papers is that they are only focused on the recommendation of individual tasks or steps. Based on the structure collected in Darwin, we envision more sophisticated recommendations that go beyond tasks.

7.3.3. Mining of Work Templates

Mining of work templates is similar to the already mentioned recommendations. The main difference is that recommendations are within one work plan, while mining of work templates takes place after several iterations of the corresponding work plans have been carried out. Another difference is that the main stakeholders for recommendations are authors in our approach. The mining of work templates is more relevant for tailors that are responsible for the maintenance of them. In the current implementation tailors can see all structuring concepts that are created or adapted by authors. This might overwhelm tailors in case large numbers of work plans are used productively. Recurring patterns in the work plans can be revealed through existing process mining algorithms. An initial step in this direction has been proposed in [SZJ13]. In this paper process mining algorithms are applied on KiPs that are specified with CMMN, e.g., in order to identify dependencies between tasks. In the first step the paper proposes the creation of a CMMN process skeleton. In the second step instances of this process are analyzed with process mining techniques. Although the lifecycle of the skeleton is not described in detail, the concept of skeletons is very similar to work templates.

7.3.4. Standardized Experiments and Evaluation Processes

Standards for the evaluation of solutions for KiPs are an open research area, which makes the assessment and comparison of different approaches difficult. In this thesis we introduced an evaluation framework that is based on controlled software experiments. In our experiments the flexibility and structure are assessed with control groups to evaluate our main research hypothesis. Besides the structuring of KiPs through end-users there are many other requirements that have to be considered. Due to the large extent of this research area, we assume that several evaluation alternatives are conceivable. A first step for future research is the identification of standardized application scenarios with different degrees of structure, e.g., the requisition and procurement of orders presented in [Hu11b]. Based on these application scenarios common criteria for the assessment of solutions are necessary that go beyond the structuring of KiPs, e.g., integration with external systems, versioning and reporting.

Bibliography

[AB01] van der Aalst, W. M.; Berens, P.: *Beyond workflow management: product-driven case handling.* In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work.* pages 42–51. ACM. 2001.

[Ab15] Abrek, N.: *Design and Implementation of a Mobile Application for the Collaborative Structuring of Knowledge-Intensive Processes.* Master's thesis. Technische Universität München. Germany. 2015.

[Al10] Allweyer, T.: *BPMN 2.0: introduction to the standard for business process modeling.* BoD–Books on Demand. 2010.

[AN95] Aamodt, A.; Nygård, M.: *Different roles and mutual dependencies of data, information, and knowledge—an AI perspective on their integration. Data & Knowledge Engineering.* 16(3):191–222. 1995.

[ASW03] van der Aalst, W.; Stoffele, M.; Wamelink, J.: *Case handling in construction. Automation in Construction.* 12(3):303–320. 2003.

[AWG05] van der Aalst, W. M.; Weske, M.; Grünbauer, D.: *Case handling: a new paradigm for business process support. Data & Knowledge Engineering.* 53(2):129–162. 2005.

[Be73] Bell, D.: *The coming of post-industrial society: a venture in social forecasting.* Basic Books. New York. 1973. 0-465-01281-7.

[Bh07] Bhattacharya, K.; Gerede, C.; Hull, R.; Liu, R.; Su, J.: *Towards formal analysis of artifact-centric business process models.* In *Business Process Management.* pages 288–304. Springer. 2007.

[Bi12] Birney, E.: *The making of ENCODE: lessons for big-data projects. Nature.* 489(7414):49–51. 2012.

[Bi14] Bigontina, M.: *Leveraging Artefact-Oriented Requirements Engineering through a Software System for Knowledge-Intensive Processes.* Master's thesis. Technische Universität München. Germany. 2014.

[Bl13] Bleibinhaus, S.: *Architectural Design and Implementation of a Web Application for Adaptive Data Models.* Master's thesis. Technische Universität München.

Germany. 2013.

[BM11] Brynjolfsson, E.; McAfee, A.: *Race against the machine: How the digital revolution is accelerating innovation, driving productivity, and irreversibly transforming employment and the economy.* Digital Frontier Press Lexington, MA. 2011.

[Bö11] Böhringer, M.: *Emergent case management for ad-hoc processes: A solution based on microblogging and activity streams.* In *Business Process Management Workshops.* pages 384–395. Springer. 2011.

[Br96] Brooke, J.: *SUS-A quick and dirty usability scale. Usability evaluation in industry.* 189(194):4–7. 1996.

[Br12] Bry, F.; Schaffert, S.; Vrandečić, D.; Weiand, K.: *Semantic wikis: Approaches, applications, and perspectives.* Springer. 2012.

[BS87] Beruvides, M. G.; Sumanth, D. J.: *Knowledge work: A conceptual analysis and structure. Productivity Management Frontiers-I.* pages 127–138. 1987.

[Bu08] Buckl, S.; Ernst, A. M.; Lankes, J.; Matthes, F.: *Enterprise Architecture Management Pattern Catalog (Version 1.0, February 2008).* Technical report. Chair for Informatics 19 (sebis), Technische Universität München. Munich, Germany. 2008.

[Bu12] Buschle, M.; Ekstedt, M.; Grunow, S.; Hauder, M.; Matthes, F.; Roth, S.: *Automating Enterprise Architecture Documentation using an Enterprise Service Bus.* In *18th Americas Conference on Information Systems, AMCIS 2012, Seattle, Washington August 9-11.* 2012.

[Ca15] Cano, I.; Alonso, A.; Hernandez, C.; Burgos, F.; Barberan-Garcia, A.; Roldan, J.; Roca, J.: *An adaptive case management system to support integrated care services: Lessons learned from the NEXES project. Journal of Biomedical Informatics.* 55:11–22. 2015.

[CGH09] Cleven, A.; Gubler, P.; Hüner, K. M.: *Design alternatives for the evaluation of design science research artifacts.* In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology.* page 19. ACM. 2009.

[CMV09] Clair, L. C.; Moore, C.; Vitti, R.: *Dynamic Case Management — An Old Idea Catches New Fire.* Technical report. Forrester. Cambridge, MA. 2009.

[Co90] Cooper, R. G.: *Stage-gate systems: a new tool for managing new products. Business horizons.* 33(3):44–54. 1990.

[Co94] Cooper, R. G.: *Third-generation new product processes. Journal of Product Innovation Management.* 11(1):3–14. 1994.

[Da05] Davenport, T. H.: *Thinking for a living: how to get better performances and results from knowledge workers.* Harvard Business Press. 2005.

[DCMR13] Di Ciccio, C.; Marrella, A.; Russo, A.: *Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches. Journal on Data Semantics.* pages 1–29. 2013.

[DHV13] Damaggio, E.; Hull, R.; Vaculín, R.: *On the equivalence of incremental and fixpoint semantics for business artifacts with Guard–Stage–Milestone lifecycles. Information Systems.* 38(4):561–584. 2013.

[DIZ06] Di Iorio, A.; Zacchiroli, S.: *Constrained wiki: an oxymoron?* In *Proceedings of the 2006 international symposium on Wikis.* pages 89–98. ACM. 2006.

[DLH02] Duyne, D. K. V.; Landay, J.; Hong, J. I.: *The design of sites: patterns, principles, and processes for crafting a customer-centered Web experience.* Addison-Wesley Longman Publishing Co., Inc. 2002.

[DN94] Davenport, T.; Nohria, N.: *Case Management and the Integration of Labor. MIT Sloan Management Review.* 35(2):11–23. 1994.

[DP98] Davenport, T. H.; Prusak, L.: *Working knowledge: How organizations manage what they know.* Harvard Business Press. 1998.

[dPv10] de Man, H.; Prasad, S.; van Donge, T.: *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done.* chapter Innovation Management, pages 211–255. Meghan-Kiffer Press. Tampa, Florida, USA. 1st edition. 2010.

[Dr77] Drucker, P.: *Management.* Harper's College Press. New York. 1977. An abridged and revised version of Management: tasks, responsibilities, practices.

[Dr93] Drucker, P. F.: *The New Society: The Anatomy of Industrial Society.* Transaction publishers. 1993.

[DR09] Dadam, P.; Reichert, M.: *The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science-Research and Development.* 23(2):81–97. 2009.

[Fa09] Fahland, D.; Lübke, D.; Mendling, J.; Reijers, H.; Weber, B.; Weidlich, M.; Zugal, S.: *Declarative versus imperative process modeling languages: The issue of understandability.* In *Enterprise, Business-Process and Information Systems Modeling.* pages 353–366. Springer. 2009.

[Fa13] Farwick, M.; Breu, R.; Hauder, M.; Roth, S.; Matthes, F.: *Enterprise architecture documentation: Empirical analysis of information sources for automation.* In *System Sciences (HICSS), 2013 46th Hawaii International Conference on.* pages 3868–3877. IEEE. 2013.

[Fe06] Fettke, D.-W.-I. P.: *State-of-the-Art des State-of-the-Art. Wirtschaftsinformatik.* 48(4):257–266. 2006.

[Fe10] Fernández, D. M.; Penzenstadler, B.; Kuhrmann, M.; Broy, M.: *A meta model for artefact-orientation: fundamentals and lessons learned in requirements engineering.* In *Model Driven Engineering Languages and Systems.* pages 183–197. Springer. 2010.

[FHS13] Fiedler, M.; Hauder, M.; Schneider, A. W.: *Foundations for the Integration of Enterprise Wikis and Specialized Tools for Enterprise Architecture Management.* In *11. Internationale Tagung Wirtschaftsinformatik, Leipzig, Germany, February*

27 – March 1, 2013. page 109. 2013.

[Fi00] Fielding, R. T.: *Architectural styles and the design of network-based software architectures.* PhD thesis. University of California, Irvine. 2000.

[Ge11] Geiger, D.; Seedorf, S.; Schulze, T.; Nickerson, R. C.; Schader, M.: *Managing the Crowd: Towards a Taxonomy of Crowdsourcing Processes.* In *AMCIS.* 2011.

[Ge14] Gerstner, M.: *Empowering End-users to Support Knowledge-intensive Processes with the Case Management Model and Notation.* Master's thesis. Technische Universität München. Germany. 2014.

[Gi15a] Gil, Y.; Michel, F.; Ratnakar, V.; Hauder, M.: *Organic Data Science: A Task-Centered Interface to On-Line Collaboration in Science.* In *Proceedings of the ACM International Conference on Intelligent User Interfaces.* Atlanta, GA. 2015.

[Gi15b] Gil, Y.; Michel, F.; Ratnakar, V.; Hauder, M.; Duffy, C.; Hanson, P.: *A Task-Centered Framework for Computationally-Grounded Science Collaborations.* In *11th International Conference on E-Science (e-Science).* IEEE. 2015.

[Gi15c] Gil, Y.; Michel, F.; Ratnakar, V.; Read, J.; Hauder, M.; Duffy, C.; Hanson, P. et al.: *Supporting Open Collaboration in Science through Explicit and Linked Semantic Description of Processes.* In *Proceedings of the Twelfth European Semantic Web Conference (ESWC).* Portoroz, Slovenia. 2015.

[GL10] Gläser, J.; Laudel, G.: *Experteninterviews und qualitative Inhaltsanalyse.* Springer-Verlag. 2010.

[GRA08] Guenther, C. W.; Reichert, M.; van der Aalst, W. M.: *Supporting Flexible Processes with Adaptive Workflow and Case Handling.* In *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.* pages 229–234. IEEE. 2008.

[Ha13a] Hauder, M.: *Bridging the gap between social software and business process management: A research agenda: Doctoral consortium paper.* In *Seventh International Conference on Research Challenges in Information Science (RCIS).* pages 1–6. IEEE. 2013.

[Ha13b] Hauder, M.; Roth, S.; Matthes, F.; Lau, A.; Matheis, H.: *Supporting collaborative product development through automated interpretation of artifacts.* In *3rd International Symposium on Business Modeling and Software Design.* 2013.

[Ha14a] Hauder, M.; Münch, D.; Michel, F.; Utz, A.; Matthes, F.: *Examining Adaptive Case Management to Support Processes for Enterprise Architecture Management.* In *Enterprise Distributed Object Computing Conference Workshops (EDOCW).* IEEE. 2014.

[Ha14b] Hauder, M.; Roth, S.; Schulz, C.; Matthes, F.: *Agile enterprise architecture management: an analysis on the application of agile principles.* In *International Symposium on Business Modeling and Software Design (BMSD).* 2014.

[He04] Hevner, A. R.; March, S. T.; Park, J.; Ram, S.: *Design Science in Information Systems Research. MIS Quarterly.* 28(1):75–105. 2004.

[HFH15] Huber, S.; Fietta, M.; Hof, S.: *Next step recommendation and prediction based on process mining in adaptive case management.* In *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management.* page 3. ACM. 2015.

[HGL11] Hauder, M.; Gil, Y.; Liu, Y.: *A framework for efficient data analytics through automatic configuration and customization of scientific workflows.* In *E-Science (e-Science), 2011 IEEE 7th International Conference on.* pages 379–386. IEEE. 2011.

[HJD10] Hull, E.; Jackson, K.; Dick, J.: *Requirements engineering.* Springer Science & Business Media. 2010.

[HKM15] Hauder, M.; Kazman, R.; Matthes, F.: *Empowering End-Users to Collaboratively Structure Processes for Knowledge Work.* In *Business Information Systems.* pages 207–219. Springer. 2015.

[HL01] Hofmann, H. F.; Lehner, F.: *Requirements engineering as a success factor in software projects. IEEE software.* 18(4):58–66. 2001.

[HPM14] Hauder, M.; Pigat, S.; Matthes, F.: *Research challenges in adaptive case management: A literature review.* In *18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW).* pages 98–107. IEEE. 2014.

[HS11] Hauschildt, J.; Salomo, S.: *Innovationsmanagement.* Vahlen. 2011.

[Hu99] Hull, R.; Llirbat, F.; Simon, E.; Su, J.; Dong, G.; Kumar, B.; Zhou, G.: *Declarative workflows that support easy modification and dynamic browsing.* In *WACC.* pages 69–78. ACM. 1999.

[Hu05] Hube, G.: *Beitrag zur Analyse und Beschreibung der Wissensarbeit. Heimsheim: Jost-Jetter.* 2005.

[Hu08] Huang, A.: *Similarity measures for text document clustering.* In *Proceedings of the sixth new zealand computer science research student conference, Christchurch, New Zealand.* pages 49–56. 2008.

[Hu11a] Hull, R.; Damaggio, E.; De Masellis, R.; Fournier, F.; Gupta, M.; Heath III, F. T.; Hobson, S. et al.: *Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events.* In *Proceedings of the 5th ACM international conference on Distributed event-based system.* pages 51–62. ACM. 2011.

[Hu11b] Hull, R.; Damaggio, E.; Fournier, F.; Gupta, M.; Heath, III, F. T.; Hobson, S.; Linehan, M. et al.: *Introducing the Guard-stage-milestone Approach for Specifying Business Entity Lifecycles.* In *Proceedings of the 7th International Conference on Web Services and Formal Methods.* WS-FM'10. pages 1–24. Berlin, Heidelberg. 2011. Springer-Verlag.

[Hu11c] Hull, R.; Damaggio, E.; Fournier, F.; Gupta, M.; Heath III, F. T.; Hobson, S.; Linehan, M. et al.: *Introducing the guard-stage-milestone approach for specifying business entity lifecycles.* In *Web services and formal methods.* pages 1–24.

Springer. 2011.

[IE05] IEEE: *Standard for Application and Management of the Systems Engineering Process. IEEE Std 1220-2005.* pages 1–87. 2005.

[IS01] ISO/IEC: *ISO/IEC 9126. Software engineering – Product quality.* ISO/IEC. 2001.

[Ke08] Kerremans, M.: *Case Management Is a Challenging BPMS Use Case.* Technical Report December. Gartner. Stamford, CT. 2008. Gartner research number G00162739.

[KH12] Kurz, M.; Herrmann, C.: *Adaptive Case Management–Anwendung des Business Process Management 2.0-Konzepts auf schwach strukturierte Geschäftsprozesse. Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg.* 2012.

[KR11] Künzle, V.; Reichert, M.: *PHILharmonicFlows: Research and Design Methodology.* Technical Report UIB-2011-05. University of Ulm. University of Ulm. May 2011.

[Kr12a] Kraut, R. E.; Resnick, P.; Kiesler, S.; Burke, M.; Chen, Y.; Kittur, N.; Konstan, J. et al.: *Building successful online communities: Evidence-based social design.* Mit Press. 2012.

[KR12b] Künzle, V.; Reichert, M.: *Striving for object-aware process support: How existing approaches fit together.* In *Data-Driven Process Discovery and Analysis.* pages 169–188. Springer. 2012.

[Kr14] Kreher, U.: *Konzepte, Architektur und Implementierung adaptiver Prozessmanagementsysteme.* PhD thesis. University of Ulm. 2014.

[KRM06] Kaan, K.; Reijers, H.; van der Molen, P.: *Introducing Case Management: Opening Workflow Management's Black Box.* In *Business Process Management.* volume 4102 of *Lecture Notes in Computer Science.* pages 358–367. Springer Berlin Heidelberg. 2006.

[Ku03] Kumaran, S.; Nandi, P.; Heath, T.; Bhaskaran, K.; Das, R.: *ADoc-Oriented Programming.* In *Proceedings of the 2003 Symposium on Applications and the Internet.* SAINT '03. pages 334–341. Washington, DC, USA. 2003. IEEE Computer Society.

[Kü13] Künzle, V.: *Object-aware Process Management.* PhD thesis. University of Ulm. 2013.

[Ku15] Kurz, M.; Schmidt, W.; Fleischmann, A.; Lederer, M.: *Leveraging CMMN for ACM: examining the applicability of a new OMG standard for adaptive case management.* In *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management.* ACM. 2015.

[KWR11] Künzle, V.; Weber, B.; Reichert, M.: *Object-aware business processes: Fundamental requirements and their support in existing approaches. International Journal of Information System Modeling and Design (IJISMD).* 2(2):19–46. 2011.

[Le51] Lewin, K.: *Field theory in social science.* Harper. New York. 1951.

[Li14] Link, P.: *Agile Methoden im Produkt-Lifecycle-Prozess–Mit agilen Methoden die Komplexität im Innovationsprozess handhaben.* In *Komplexitätsmanagement in Unternehmen.* pages 65–92. Springer. 2014.

[LKL10] Lucke, C.; Krell, S.; Lechner, U.: *Critical Issues in Enterprise Architecting - A Literature Review.* In *Proceedings of the Sixteenth Americas Conference on Information Systems (AMCIS 2010).* Lima, Peru. 2010.

[LW04] Langenberg, K.; Wegmann, A.: *Enterprise Architecture: What Aspects is Current Research Targeting?* Technical report. EPFL Switzerland. 2004.

[Ma62] Machlup, F.: *The production and distribution of knowledge in the United States.* volume 278. Princeton university press. 1962.

[MB15] Marin, M. A.; Brown, J. A.: *Implementing a Case Management Modeling and Notation (CMMN) System using a Content Management Interoperability Services (CMIS) compliant repository. arXiv preprint arXiv:1504.06778.* 2015.

[MBR15] Mundbrod, N.; Beuter, F.; Reichert, M.: *Supporting Knowledge-intensive Processes Through Integrated Task Lifecycle Support.* In *19th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2015).* IEEE Computer Society Press. July 2015.

[Mc10] McCauley, D.: *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done.* chapter Achieving Agility, pages 257–275. Meghan-Kiffer Press. Tampa, Florida, USA. 1st edition. 2010.

[MF11a] Marjanovic, O.; Freeze, R.: *Knowledge intensive business processes: theoretical foundations and research challenges.* In *System Sciences (HICSS), 2011 44th Hawaii International Conference on.* pages 1–10. IEEE. 2011.

[MF11b] Mendez Fernandez, D.; Lochmann, K.; Penzenstadler, B.; Wagner, S.: *A case study on the application of an artefact-based requirements engineering approach.* In *Evaluation Assessment in Software Engineering (EASE 2011), 15th Annual Conference on.* pages 104–113. April 2011.

[MHM15] Marin, M. A.; Hauder, M.; Matthes, F.: *Case Management: An Evaluation of Existing Approaches for Knowledge-Intensive Processes.* In *4th International Workshop on Adaptive Case Management and other non-workflow Approaches to BPM.* 2015.

[MHV13] Marin, M.; Hull, R.; Vaculín, R.: *Data centric bpm and the emerging case management standard: A short survey.* In *Business Process Management Workshops.* pages 24–30. Springer. 2013.

[Mi15a] Michel, F.: *A Structured Task-Centered Framework for Online Collaboration.* Master's thesis. Technische Universität München. Germany. 2015.

[Mi15b] Michel, F.; Gil, Y.; Ratnakar, V.; Hauder, M.: *A Virtual Crowdsourcing Community for Open Collaboration in Science Processes.* In *21st Americas Conference* *on Information Systems (AMCIS).* AIS. 2015.

[MKR13] Mundbrod, N.; Kolb, J.; Reichert, M.: *Towards a system support of collaborative knowledge work.* In *Business Process Management Workshops.* pages 31–42. Springer. 2013.

[MLVDP14] Marin, M. A.; Lotriet, H.; Van Der Poll, J. A.: *Measuring Method Complexity of the Case Management Modeling and Notation (CMMN).* In *Proceedings of the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology.* SAICSIT '14. pages 209:209–209:216. New York, NY, USA. 2014. ACM.

[MN11] Matthes, F.; Neubert, C.: *Wiki4eam: Using hybrid wikis for enterprise architecture management.* In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration.* pages 226–226. ACM. 2011.

[MNB11] Motahari-Nezhad, H. R.; Bartolini, C.: *Next best step and expert recommendation for collaborative processes in it service management.* In *Business Process Management.* pages 50–61. Springer. 2011.

[MNS11] Matthes, F.; Neubert, C.; Steinhoff, A.: *Hybrid Wikis: Empowering Users to Collaboratively Structure Information.* In *ICSOFT (1).* pages 250–259. 2011.

[MNS13] Motahari-Nezhad, H. R.; Swenson, K. D.: *Adaptive Case Management: Overview and Research Challenges.* In *15th Conference on Business Informatics (CBI).* pages 264–269. IEEE. 2013.

[Mo09] Moser, C.; Junginger, S.; Brückmann, M.; Schöne, K.-M.: *Some Process Patterns for Enterprise Architecture Management.* In *Software Engineering (Workshops).* pages 19–30. Citeseer. 2009.

[MR14] Mundbrod, N.; Reichert, M.: *Process-Aware Task Management Support for Knowledge-Intensive Business Processes: Findings, Challenges, Requirements.* In *Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International.* pages 116–125. IEEE. 2014.

[Mu12] Mundbrod, N.: *Business Process Support for Collaborative Knowledge Workers.* Master's thesis. Ulm University. 2012.

[MZ90] Mathieu, J. E.; Zajac, D. M.: *A review and meta-analysis of the antecedents, correlates, and consequences of organizational commitment. Psychological bulletin.* 108(2):171. 1990.

[Ne12] Neubert, C.: *Facilitating Emergent and Adaptive Information Structures in Enterprise 2.0 Platforms.* Dissertation. Technische Universität München. München. 2012.

[Ni94] Nielsen, J.: *Usability engineering.* Elsevier. 1994.

[Ni12] Nielsen, M.: *Reinventing discovery: the new era of networked science.* Princeton University Press. 2012.

[No94] Nonaka, I.: *A dynamic theory of organizational knowledge creation. Organization science.* 5(1):14–37. 1994.

[NT95] Nonaka, I.; Takeuchi, H.: *The knowledge-creating company: How Japanese companies create the dynamics of innovation.* Oxford university press. 1995.

[Oa04] Odersky, M.; al.: *An Overview of the Scala Programming Language.* Technical Report IC/2004/64. EPFL Lausanne, Switzerland. 2004.

[Ob14] Object Management Group (OMG): *Case Management Model and Notation (CMMN), Version 1.0 (formal/2014-05-05).* `http://www.omg.org/spec/CMMN/`. 2014.

[Pa10] Palmer, N.: *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done.* chapter Introduction, pages 1–4. Meghan-Kiffer Press. Tampa, Florida, USA. 1st edition. 2010.

[PCR06] Peter, R.; Coronel, C.; Rob, P.: *Database Systems: Design, Implementation, and Management.* Crisp Learning. 2006.

[Pe07] Peffers, K.; Tuunanen, T.; Rothenberger, M. A.; Chatterjee, S.: *A design science research methodology for information systems research. Journal of management information systems.* 24(3):45–77. 2007.

[Pe08] Pesic, M.: *Constraint-Based Workflow Management Systems: Shifting Control to Users.* PhD thesis. Eindhoven University of Technology. 2008.

[Po66] Polanyi, M.: *The Tacit Dimension.* Doubleday. New York. 1966.

[Po11] Porter, M. E.: *Competitive advantage of nations: creating and sustaining superior performance.* Simon and Schuster. 2011.

[RB96] Rossi, M.; Brinkkemper, S.: *Complexity metrics for systems development methods and techniques. Information Systems.* 21(2):209–227. 1996.

[Re88] Resch, M.: *Die Handlungsregulation geistiger Arbeit: Bestimmung und Analyse geistiger Arbeitstätigkeiten in der industriellen Produktion.* Huber. 1988.

[Re09] Redding, G.; Dumas, M.; Ter Hofstede, A. H.; Iordachescu, A.: *Modelling flexible processes with business objects.* In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on.* pages 41–48. IEEE. 2009.

[Ro03] Ross, J. W.: *Creating a strategic IT architecture competency: Learning in stages. MIS Quarterly Executive.* 2(1):31–43. 2003.

[Ro13] Roth, S.; Hauder, M.; Farwick, M.; Breu, R.; Matthes, F.: *Enterprise Architecture Documentation: Current Practices and Future Directions.* In *Wirtschaftsinformatik.* page 58. 2013.

[RRA03] Reijers, H. A.; Rigter, J.; van der Aalst, W. M.: *The case handling case. International Journal of Cooperative Information Systems.* 12(03):365–391. 2003.

[RWR06] Ross, J. W.; Weill, P.; Robertson, D.: *Enterprise architecture as strategy: Creating a foundation for business execution.* Harvard Business Press. 2006.

[Sc99] Scarbrough, H.: *Knowledge as work: conflicts in the management of knowledge workers. Technology analysis & strategic management.* 11(1):5–16. 1999.

[SM95] Strong, D. M.; Miller, S. M.: *Exceptions and exception handling in computerized information processes. ACM Transactions on Information Systems (TOIS).* 13(2):206–233. 1995.

[Sw10a] Swenson, K. D.: *Mastering the unpredictable: How adaptive case management will revolutionize the way that knowledge workers get things done.* Meghan-Kiffer Press. 2010.

[Sw10b] Swenson, K. D.: *Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done.* chapter The Nature of Knowledge Work, pages 5–28. Meghan-Kiffer Press. Tampa, Florida, USA. 1st edition. 2010.

[Sw13] Swenson, K. D.: *Designing for an innovative learning organization.* In *Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International.* pages 209–213. IEEE. 2013.

[SZJ13] Schonig, S.; Zeising, M.; Jablonski, S.: *Supporting collaborative work by learning process models and patterns from cases.* In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on.* pages 60–69. IEEE. 2013.

[Th09] The Open Group: *TOGAF, Version 9.1.* `http://www.opengroup.org/subjectareas/enterprise/togaf/`. 2009.

[Ut14] Utz, A.: *Empowering Users to Collaboratively Structuring Innovation Processes.* Master's thesis. Technische Universität München. Germany. 2014.

[Va11] Vaculin, R.; Hull, R.; Heath, T.; Cochran, C.; Nigam, A.; Sukaviriya, P.: *Declarative business artifact centric modeling of decision and knowledge intensive business processes.* In *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International.* pages 151–160. IEEE. 2011.

[We12] Weske, M.: *Business process management: concepts, languages, architectures.* Springer Science & Business Media. 2012.

[Wh09] White, M.: *Case management: Combining Knowledge with Process. BPTrends, July.* 2009.

[Wo12] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A.: *Experimentation in software engineering.* Springer Science & Business Media. 2012.

[WW02] Webster, J.; Watson, R. T.: *Analyzing the past to prepare for the future: Writing a literature review. Management Information Systems Quarterly.* 26(2):13–23. 2002.

[Yi15] Yiqin, Y.; Xiang, L.; Haifeng, L.; Jing, M.; Nirmal, M.; Vatche, I.; Guotong, X. et al.: *Case Analytics Workbench: Platform for Hybrid Process Model Creation and Evolution.* In *13th International Conference on Business Process Management.* 2015.

[ZMR08] Zur Muehlen, M.; Recker, J.: *How much language is enough? Theoretical and practical use of the business process modeling notation.* In *Advanced information* *systems engineering.* pages 465–479. Springer. 2008.

APPENDIX A

API Methods

A.1. Login to the system

URL
: /api/login

Method
: POST

Controller
: controllers.auth.AuthenticationController.login()

Response
: status of the login request, e.g., status code 403 when login failed

Users can login with their e-mail and password in the server that responses with a successful login in case the credential could be verified. Otherwise the server responses with http status code 403 for forbidden and any access to the server is prevented. After a successful login the client is automatically redirected to the start page which is the social feed that is introduced in Section 3.4.1. Thereby, the social feed is the central point for communication in the software solution. Once users are logged in they immediately see changes that occurred since their last login in the system. The login uses a module for authorization and authentication[1] that is available for the play framework. The module provides several interfaces that need to be implemented.

[1] `https://github.com/t2v/play2-auth/tree/release0.10.1`, last accessed: 2015-06-07

A.2. Logout of the System

URL
: /api/logout

Method
: POST

Controller
: controllers.auth.AuthenticationController.logout()

Response
: status of the logout request

Users are logged out from the server with this request that is forwarded to the authentication controller. After the logout users are automatically redirected to the login page. The logout uses the same module for authorization and authentication that is provided for the play framework like the login. In the current implementation users that are logged out are not able to get any access to the system through the API, since this was not necessary for the purpose of the research conducted in this thesis. In future work, content that is not confidential in the server could be provided to users that are not logged in. Alternatively, the server could expose this public data for other systems through its API.

A.3. Get All Pages and Wikis

URL
: /api/explorer

Method
: GET

Controller
: controllers.index.ExplorerController.get()

Response
: names and ids of all wikis with their pages and subpages

The data structure for the explorer returns all wikis and pages that are stored in the database with their label and id. Subsequent requests can use this data structure to navigate through the entities in the system. The label contains the name and the id which is a unique ObjectId to identify the entity. The kind of entity is determined using an additional attribute called class, which can be either wiki page, wiki, or a type. Every wiki and wiki page also contains an attribute to determine whether it is starred and watched by the logged in user. Starred wikis and wiki pages are shown in the favorites of the user and these values are stored separately in the data structure for every user. This method is used to create the explorer that is shown on the left in the user interface.

A.4. Creation of a New Wiki

URL
: /api/wiki

Method
: PUT

Controller
: controllers.wiki.WikiController.create()

Response
: id of the new wiki page that is created

Creates a new wiki in the database with the name that has been specified in the request, whereas only users that are logged in are able to perform this operation. Information about the user who initiated this request is necessary to create a history entry that captures the information when the wiki was created by whom. For the wiki no type or attributes are provided since it is only used as container for wiki pages as described in Section 3.2. The method returns the unique ObjectId of the new wiki as response in case the creation of the wiki was successful in the database. After this operation the user interface is reloaded by an AngularJS controller and the previously described request for the explorer is used to gather all wikis with their pages, so that the new wiki appears in the explorer.

A.5. Find Wiki by Id

URL
: /api/wiki/:wikiId

Method
: GET

Controller
: controllers.wiki.WikiController.get(wikiId: ObjectId)

Response
: data structure of the wiki with the id wikiId or 404 when not found

The wiki with the id wikiId is searched by the unique id in the database. This request can be used in combination with the explorer request, i.e., the explorer provides all wikis and this request is used to get the data structure of one wiki. In case no wiki with this wikiId is available in the database, the controller returns not found with http status code 404. Otherwise the entire data structure of the wiki consisting of the name, text on the wiki and history information is returned by the server. The history information can be used by different frond-ends to show when the wiki was created and edited by whom, e.g., for the social feed feature. Navigation to wiki pages within this wiki is possible with the data structure that is provided by the explorer request.

A.6. Delete Wiki

URL
: /api/wiki/:wikiId

Method
: DELETE

Controller
: controllers.wiki.WikiController.delete(wikiId: ObjectId)

Response
: status code 200 ok when deleted or 404 when not found

The deletion of an entire wiki is a two step process in case the wiki with the given wikiId exists in the system. If no wiki with this wikiId is existing in the database, the method returns the status code not found. In the first step all pages, types and tasks within this wiki are recursively deleted. In this step references of this wiki that are used for the favorite functionality are also removed for the users. These references are stored in the data structure of the users. In the second step the wiki with the given wikiId is deleted from the database. In the current implementation versioning is not supported, i.e., it is not possible to recover wikis with their content once they are deleted. Due to the ramifications of this request users should receive a warning message that requires an additional approval before the controller is executed.

A.7. Update Text of a Wiki

URL
: /api/wiki/:wikiId/update/text

Method
: POST

Controller
: controllers.wiki.WikiController.updateText(wikiId: ObjectId)

Response
: status code 200 ok when updated or 404 when not found

This method updates the text that is shown as content for the wiki with the ObjectId wikiId. The server responds with not found in case the wikiId is not available in the database or with ok and status code 200 if the operation was successful. During this operation the last editor of the wiki text is also updated with the user that triggered the operation. After the text has been updated the controller automatically generates a history event to describe when the text was updated by whom.

A.8. Create New Wiki Page

URL
: /api/page

Method
: PUT

Controller
: controllers.page.PageController.create()

Response
: pageId of the new wiki page

Creates a new wiki page with the provided name in an existing wiki that already has to be available in the system. It is possible to specify a parent page to add the new page below an existing page in the hierarchy. In case no parent page is specified, the new page is located at the highest level in the wiki. Another optional parameter is the type of the wiki page that can be specified during the creation. The type can be either new in the system or it might be based on an existing TypeDefinition, i.e., in the latter case this corresponds to the instantiation of an existing work template.

A.9. Get a Wiki Page

URL
: /api/page/:pageId

Method
: GET

Controller
: controllers.page.PageController.get(pageId: ObjectId)

Response
: entire data structure of the page

The data structure of the page with the provided pageId is returned. This method is called every time the page with the pageId is shown in the browser. It returns all attributes and tasks assigned to the page, whereas the tasks are divided into completed and available tasks. Part of this data structure are also the text, type and name of the wiki page. The progress of the page is also part of the data structure to avoid unnecessary computations if the progress has not changed. In case the user selected a task on this page, this task is stored in the database as current task for the page and returned with this method together with the values for the three roles execute, delegate and skip.

A.10. Update Text of the Wiki Page

URL
: /api/page/:pageId/update/text

Method
: POST

Controller
: controllers.page.PageController.updateText(pageId: ObjectId)

Response
: status of the update for the wiki page, e.g., 200 when successful

The page with the pageId is retrieved from the database and in case it is not available not found is returned as status code. The new text of the wiki page is updated and the page is stored in the database. After the update of the wiki page text the method creates a history entry for this operation, which basically contains the information when the text was updated by whom.

A.11. Add an Attribute to the Wiki Page

URL
: /api/page/:pageId/attribute

Method
: PUT

Controller
: controllers.page.PageController.addAttribute(pageId: ObjectId)

Response
: status of the add attribute operation, e.g., 400 for bad request

This method adds a new attribute to the wiki page with the id pageId. The method requires some input parameters so that the attribute can be created, otherwise the method returns the http status code for bad request. It requires the name of the attribute, type and access rights. With this information a new attribute can be created and added to the provided page with the id pageId. The combination of name and attribute type needs to be unique for one users, whereas different users could create attributes with the same type and name if the access rights are different for them. This might be necessary if the same attribute already exists, but the user has no read access for it. In our approach tailors need to evaluate the reason for redundant attributes, e.g., it might be necessary to change the access rights. After the page with this id is stored in the database with the new attribute, the controller generates a page history event with the information when the attribute was added to the page by whom.

A.12. Delete Page

URL
: /api/page/:pageId

Method
: DELETE

Controller
: controllers.page.PageController.delete(pageId: ObjectId)

Response
: status of the page deletion, e.g., 200 when successful

This method deletes the wiki page with the unique id pageId from the database. Similar to the deletion of an entire wiki this methods deletes a wiki page in two steps. In the first step all children pages of the page with the given pageId are removed if they have no other parent pages assigned. Children pages that have other parent pages assigned are not deleted from the database and the pageId is removed from the association to parent pages. In the second step all associations of the given page are removed. The associated types and tasks are removed as well, i.e., we perform cascading deletes.

A.13. Delete an Attribute of the Wiki Page

URL
: /api/page/:pageId/attribute

Method
: DELETE

Controller
: controllers.page.PageController.deleteAttribute(pageId: ObjectId)

Response
: status of the delete attribute operation, e.g., 200 when successful

The attribute with the given name and type is deleted from the page with the pageId. The combination of an attribute name and type is unique for one users so that this is enough information to determine the attribute to delete. Deleting an attribute from a wiki page requires several steps since attributes can be associated with tasks. First, the task that is associated to the attribute is determined and the attribute is removed from this task. In case the attribute is referenced to another page with subtasks, this is considered for the progress of the associated task, i.e., the progress of the task is updated without the subtasks. After successful completion of the operation, the controller generates a page history event with the information when the attribute was deleted by whom. Another method in the API is provided for the update of an attribute that is not explained.

A.14. Add Value for an Attribute

URL
: /api/page/:pageId/attribute/value

Method
: PUT

Controller
: controllers.page.PageController.addAttributeValue(pageId: ObjectId)

Response
: status of the delete attribute operation, e.g., 200 when successful

In case the attribute is not existing on the wiki page, it is automatically created by the controller. If the attribute is assigned to a task on the wiki page, the progress of the task and the associated page is updated as described in Section 3.2.3. Depending on the value that is added to the attribute this operation will update the progress of the assigned tasks. In some cases it might be possible that the assigned task is completed, i.e., only one value is missing for the task. In any case the controller generates an event in the page history for the adding of the value. If a task is assigned to the attribute with the new value, another page history entry is added for the changing progress of the task. Additional controller to update and delete attribute values are provided that are not explained for the sake of brevity.

A.15. Create New Task on a Wiki Page

URL
: /api/page/:pageId/task

Method
: PUT

Controller
: controllers.page.PageController.newTask(pageId: ObjectId)

Response
: status 200 for ok or internal server error with status code 500

This method creates a new task for a wiki page and is triggered by the task representation described in Section 3.3.1. In the first step the wiki page with the unique pageId is gathered from the database. The new task with the specified name is created and added to the wiki page. After this operation the controller generates a history event to describe when the task was created by whom on the wiki page. The progress of the wiki page is updated since the page progress needs to be changed, i.e., the overall progress of parent tasks and the wiki page becomes lower since every new task has a progress of 0%.

A.16. Assign Attribute to Task on a Wiki Page

URL
: /api/page/:pageId/task/attribute

Method
: POST

Controller
: controllers.page.PageController.addTaskAttribute(pageId: ObjectId)

Response
: status 200 for ok when the attribute could be assigned to the task

A new attribute is added to the task on the wiki page with the unique id pageId as mandatory work result for this task. For this purpose the request provides the id of the task as well as the name and type of the attribute. Based on this information the page, attribute and task are gathered from the database. The controller creates a copy of the attribute and assigns this copy to the task. After the attribute is added, the controller updates the progress of related tasks to identify new overdue or inconsistent subtasks that might arise due to subtasks on the assigned attribute. Finally, the page progress is updated since the progress of the task with the new attribute might have changed.

A.17. Remove Attribute from Task on a Wiki Page

URL
: /api/page/:pageId/task/attribute

Method
: DELETE

Controller
: controllers.page.PageController.deleteTaskAttribute(pageId: ObjectId)

Response
: status 200 for ok when the attribute could be deleted

The request provides the id of the task as well as the name and type of the attribute that should be deleted. Based on this information and the pageId the task, attribute and page are gathered from the database. The copy of the attribute is deleted from the data structure of the task. Similar to the assignment of an attribute to a task, the controller updates the progress of related tasks. This is necessary since existing overdue or inconsistent subtasks might have been removed and the state of the tasks needs to be changed according to the task state sequence described in Section 3.3.4. Finally, the page progress is updated since the task progress might have changed.

A.18. Skip Task on Wiki Page

URL
: /api/page/:pageId/task/current/skip

Method
: POST

Controller
: controllers.page.PageController.skipCurrentTask(pageId: ObjectId)

Response
: status 200 for ok when the task could be skipped

This method performs the skip operation for the current task. Before this request can be executed, the task needs to be enabled as current task by clicking on it in the task representation (cf. Section 3.3.1). This request can only be triggered when the user has the skip role for this task assigned. The controller generates a history event to describe when this task has been skipped by whom. After this operation the progress of related tasks is updated since the overdue or inconsistent task might be affected by the skipping. Finally, the progress of the page is updated because the task is removed from the computation of the progress.

A.19. Update Dates of a Task

URL
: /api/page/:pageId/task/current/metaData/updateDates

Method
: POST

Controller
: controllers.page.PageController.updateDatesOfCurrentTask(pageId: ObjectId)

Response
: status 200 for ok when the dates could be changed

The request provides the new dates for the current task on the wiki page with the unique id pageId. It is possible to specify start and end dates or only one of them. Before this request can be triggered the user has to enable the task in the task representation (cf. Section 3.3.1). After the metadata for the dates are updated, the controller stores the task in the database. In the current implementation no history event is generated for this event. This method currently has no influence on the progress of a task since the progress is computed independently from the task duration (cf. Section 3.2.3). It could be possible that the update of the start and end dates changed the state of the task, i.e., an inconsistency might have been created with start or end dates that exceed the dates of the parent task. It could also be possible that an existing inconsistency has been fixed. Therefore, the progress information of related tasks are updated by the controller.

A.20. Delegate Task on Wiki Page

URL
: /api/page/:pageId/task/current/delegate

Method
: POST

Controller
: controllers.page.PageController.delegateCurrentTask(pageId: ObjectId)

Response
: status 200 for ok when the task could be delegated

The request to delegate a task on a wiki page needs to provide the user or group that should receive the task for the new execution role. To delegate a task the user has to enable the task in the task representation (cf. Section 3.3.1). This method can be executed arbitrarily often, i.e., users with delegated tasks are able to delegate them again to other users or groups. After the task is stored in the database the controller generates a history event that captures when the task was delegated by whom. The overall progress of the task and page remains unmodified, i.e., delegated tasks retain their progress.

A.21. Update Progress of a Task

URL
: /api/page/:pageId/task/current/metaData/updateProgress

Method
: POST

Controller
: controllers.page.PageController.updateProgressOfCurrentTask(pageId: ObjectId)

Response
: status 200 for ok when the progress could be changed

Users are able to update the progress of a task manually in the task metadata based on this request (cf. Section 3.3.2). The request needs to provide the new progress as integer and the unique pageId of the wiki page with the task. Before the request can be executed the user needs to enable the task that needs to be updated in the task representation. In case the progress is set to 100% it is finished in this controller, which basically means that it is shown as completed task in the task representation and the profile of the responsible user. After the updated task is stored, the controller generates a history event to capture when the progress was changed by whom and what the new progress of the task is. Similar to the aforementioned methods, the progress of the page must be updated with the new progress.

A.22. Update Expertises of a Task

URL
: /api/page/:pageId/task/current/metaData/updateExpertises

Method
: POST

Controller
: controllers.page.PageController.updateExpertises(pageId: ObjectId)

Response
: status 200 for ok when the expertise could be updated

This method either adds a new expertise to the task or removes an existing one. Therefore, the request needs to provide the string for the expertise and the unique pageId of the wiki page with the task. The controller checks whether the expertise is already existing in the metadata of the task. If it is not existing the controller adds it to the metadata of task, otherwise it is removed. In both cases the task with the new metadata is stored in the database. This controller generates no history event since this operation must not be shown in the social feed of our software solution.

A.23. Get Type Definition

URL
: /api/typeDefinition/:typeDefinitionId

Method
: GET

Controller
: controllers.wiki.TypeController.get(typedDefinitionId: ObjectId)

Response
: the data structure of the type definition

While the previous methods are concerned with concepts that are related to work plans, this method provides the type definition of a work plan. The request needs to provide the unique id typeDefinitionId that is used to gather the type definition from the database. The method returns the entire data structure that is related to the type definition. This includes the name and pageId of all wiki pages that have a type that is associated with this type definition in the wiki. All attribute definitions and attributes created by end-users that are associated are also part of this data structure. Since every type definition is unique for the wiki, the id and name of the wiki are provided. Finally, all elements related to the process structure of the type definition are provided in the data structure. The API also provides the methods required to create and delete type definitions that are not explained for the sake of brevity.

A.24. Add Attribute Definition to Type Definition

URL
: /api/typeDefinition/:typeDefinitionId/attributeDefinition

Method
: PUT

Controller
: controllers.wiki.TypeController.addAttributeDefinition(typedDefinitionId: ObjectId)

Response
: status 200 for ok or bad request with status 400

With this request a new attribute definition is added for the type definition with the unique id typeDefinitionId. Only tailors for this type definition are able to perform this request. The request needs to provide name, type and access rights for the attribute definition. The controller stores the attribute definition in the database and associates it with the type definition. In this controller no history event is generated since this method is not performed by authors. The API also provides the methods to delete and update attribute definitions that are not explained for the sake of brevity.

A.25. Add Task Definition to Type Definition

URL
: /api/typeDefinition/:typeDefinitionId/taskDefinition

Method
: PUT

Controller
: controllers.wiki.TypeController.addTaskDefinition(typedDefinitionId: ObjectId)

Response
: status 200 for ok when the task definition could be added

Tailors can use this request to add a task definition for the type definition with the unique id typeDefinitionId. The request requires the name, attributes that are assigned and the users and groups that are defined for the roles of this task definition. Similar to the adding of an attribute definition, the controller generates no history event since this method is not performed by authors. The API provides several other methods that are not explained in this thesis. Among them are methods to delete task definitions as well as to create stages and rules for the process structure. The interactive CMMN workbench that is explained in the subsequent section also requires some methods to set the position of the shapes in the process editor.

A.26. Dynamic Loading of History Entries for the Social Feed

URL
: /api/user/feed/:numberNewItems/:numberMinItems/:filter/:fromDate

Method
: GET

Controller
: controllers.user.UserController.getNextHistoryEntries(numberNewItems: Int, numberMinItems: Int, fromDate: Long, filter: String)

Response
: data structure with the entries for the social feed

Activities of authors in the system constantly generate history entries in the system. Due to the amount of data this method provides a subset of all entries of the social feed. The entries in the subset are determined by the parameters provided within the request. In the first step, the controller gathers all history entries from the database sorted by date. The number of new items parameter defines how many new items for a user have been generated since the last request. The second parameter defines how many history entries are at least loaded by the controller. The filter contains the currently applied filter on the social feed to make sure that enough history entries are shown for the selected filters. Additional methods are provided by the API for the social interaction with the history entries, e.g., commenting a history entry.

A.27. Get the Profile of User

URL
: /api/user/:userId/profile

Method
: GET

Controller
: controllers.user.UserController.profile(userId: ObjectId)

Response
: data structure of the user profile

The profile contains various information about a user and this request provides the required data for the user rating and skills (cf. Section 3.3.8) as well as the personal worklist (cf. Section 3.3.9). The controller gathers all tasks from the database that have the user with the unique id userId assigned to the execution role or that are delegated to this user. These tasks are divided into open and completed tasks by the controller. The ranking is computed based on the relation of the number of tasks that the user completed compared to the number of tasks that the other users completed. The expertises are determined through the tasks that the user completed in the system.

Nomenclature

ACM	adaptive case management, page 38
ADM	architecture development method, page 32
API	application programming interface, page 95
BISA	business information systems analysis, page 35
BMC	Business Model Canvas, page 148
BPM	business process management, page 4
BPMN	Business Process Model and Notation, page 3
CDSS	clinical decision support system, page 173
CMMN	Case Management Model and Notation, page 3
CMS	content management system, page 42
CSCW	computer supported cooperative work, page 4
EA	enterprise architecture, page 32
EAM	enterprise erchitecture management, page 32
EAMPC	enterprise architecture management pattern catalog, page 173
ECA	event condition action, page 20
EHR	electronic health records, page 173
EPC	event-driven process chain, page 120

GSM	Guard-Stage-Milestone, page 20
ICS	integrated care services, page 173
IS	information systems, page 4
IT	information technology, page 32
IUI	intelligent user interface, page 4
JSON	JavaScript Object Notation, page 119
KiP	knowledge-intensive process, page 1
KM	knowledge management, page 16
MVC	model-view-controller, page 119
OMG	Object Management Group, page 3
RFP	request for proposal, page 3
SUS	System Usability Scale, page 165
SVG	Scalable Vector Graphics, page 119
TOGAF	The Open Group Architecture Framework, page 32
TUM	Technical University Munich, page 141
UML	Unified Modeling Language, page 35
URL	uniform resource locator, page 66