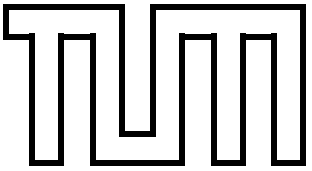


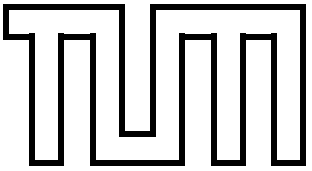
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Empowering End-users to Support Knowledge-intensive Processes with the Case Management Model and Notation

Manuel Gerstner



FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Empowering End-users to Support Knowledge-intensive Processes with the Case Management Model and Notation

Befähigung von Endanwendern zur Unterstützung von wissensintensiven Prozessen durch die Case Management Model and Notation

Author:	Manuel Gerstner
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Matheus Hauder, M.Sc.
Submission date:	December, 15 2014

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, December 15, 2014 Manuel Gerstner

Acknowledgments

I am using this opportunity to express my deepest gratitude to those people who supported me throughout the course of this Master's thesis. I am especially grateful for the constant support given by my supervisor Matheus Hauder, who provided strong guidance and useful feedback.

I would also like to thank my family and friends for their encouragement, which helped me to stay focused. Special thanks go to my sister Laura and my friend Dominique for proofreading this work.

Abstract

Next to the widespread use of workflow management solutions in practice, there are many business processes that are currently not adequately supported. These processes are often very data-driven, unstructured, unpredictable, and driven by user decisions. In the literature they are usually referred to as Knowledge-intensive Processes.

As a result the Object Management Group (OMG) has recently released the Case Management Model and Notation (CMMN) as a specification that supports the modeling of such processes. Similar to BPMN for traditional business processes this standard provides a visual notation and the operational semantics for Knowledge-intensive Processes.

Throughout the course of this master's thesis, the CMMN specification will be analyzed thoroughly in order to evaluate, whether it can support Knowledge-workers with their problems, which are often very complex and unique.

The main goal is to use the insights gained throughout this analysis to implement a subset of this notation in an existing research prototype. This includes an analysis of advantages and disadvantages of the CMMN specification as well as the selection of a subset that will be implemented based on existing workflow patterns and requirements for process modeling.

Keywords. Case Management Model and Notation, Adaptive Case Management, Knowledge-intensive Processes, Case Handling, Business Process Modeling

Zusammenfassung

Trotz der allgegenwärten Benutzung von Workflow-Management-Systemen in der Praxis, werden viele Geschäftsprozesse nur unzureichend durch die aktuellen Lösungen unterstützt. Diese Prozesse sind meist sehr datenorientiert und werden von benutzerspezifischen Entscheidungen gelenkt. Außerdem zeichnen Sie sich häufig durch ein hohes Maß an Unstrukturiertheit sowie Unvorhersehbarkeit aus. In der Literatur werden sie daher oft als wissensintensive Prozesse bezeichnet.

Aufgrund dieses Problems hat die Object Management Group (OMG) vor kurzem die Case Management Model and Notation (CMMN) veröffentlicht, welche Adaptive Case Management (ACM) unterstützen soll. Vergleichbar mit BPMN für traditionelle Geschäftsprozesse, stellt dieser neue Standard eine visuelle Notation und die operationelle Semantik für wissensintensive Prozesse bereit.

Im Verlauf dieser Masterarbeit soll die CMMN Spezifikation genau untersucht werden, um herauszufinden ob sie Wissensarbeiter bei der Lösung vielschichtiger Probleme, welche sich häufig durch ihre Komplexitt und Einzigartigkeit auszeichnen, unterstützen kann.

Das Ziel ist es, die gewonnenen Erkenntnisse für die Implementierung einer Teilmenge der Notation in einen bereits existierenden Protoypen zu benutzen. Dies beinhaltet sowohl die Analyse der Vor- und Nachteile der CMMN-Spezifikation, als auch die Auswahl einer geeigneten Teilmenge zur Implementierung. Bei der Auswahl der Teilmenge sollen bestehende Workflow Patterns und Anforderungen aus der Prozessmodellierung verwendet werden.

Schlagwörter. Case Management Model and Notation, Adaptive Case Management, wissensintensive Prozesse, Case Handling, Geschäftsprozessmodellierung

Contents

Outline of the Thesis

Part I: Processes for Knowledge Work with CMMN

CHAPTER 1: INTRODUCTION

This chapter gives an introduction to the thesis, as well as a basic overview of the context. The motivation for this thesis is explained and the concrete problems are described.

CHAPTER 2: THEORETICAL BACKGROUND

This chapter introduces the major topics discussed throughout this thesis and explains them in detail. While it starts with the most basic terminology, the specific terms important for the understanding of this work are highlighted.

CHAPTER 3: RELATED WORK

This chapter focuses on the literature and research concerning the terms introduced in Chapter 2. It summarizes the key literature with regard to the topic of this thesis.

Part II: Supporting Knowledge-intensive Processes with CMMN

CHAPTER 4: WORKFLOW PATTERNS

The existing Workflow Patterns included in most of the contemporary modeling software are analyzed thoroughly in order to generate basic requirements for the implemented prototype.

CHAPTER 5: REQUIREMENTS FOR KNOWLEDGE-INTENSIVE PROCESSES

This chapter picks up the patterns derived as requirements in the previous chapter and provides a list of requirements needed for a software implementation supporting Knowledge-intensive Processes.

the initial goals defined by the research questions have been reached.

CHAPTER 12: FUTURE OUTLOOK

The main objective of this chapter is to examine some of the areas which research can focus on using the insights gained throughout this thesis. It also identifies different aspects of the work in this thesis which require further evaluation.

Part I.

Introduction and Theory

1. Introduction

"The most valuable assets of the 20th-century company was its production equipment. The most valuable asset of a 21st-century institution will be its knowledge workers and their productivity."

Peter F. Drucker, 1999 [8]

The industrialization which has taken place over the course of the last centuries, has had a huge impact on the way people work. Processes that have been present for thousands of years were modeled, adjusted, structured and improved. This modernized not only the way people worked, but also the way in which tasks that were executed repeatedly could be brought into order and connected to a set of rules. This made such processes easy to structure and organize in a hierarchical order in a way that allowed people to model and eventually share them.

This has more recently led to the definition of modeling languages such as the Business Process Model and Notation (BPMN) which sets a common standard for the modeling of routine processes. Because of the wide use of such modeling languages in modern enterprises, the BPMN standard was further improved. Nevertheless the notation still had some disadvantages as soon as the process had certain features making them especially hard to model before execution. Such processes can mostly be described as weakly-structured and constantly changing and are often categorized as knowledge work. They require people with a certain expertise in the field as there is usually no strict process that provides them with guidance.

An example for such a process would be the design of a complex system architecture. Due to the high complexity of the task and the uniqueness of every process iteration, only people with expert knowledge are able to come up with a solution that solves the predefined problems. The major problem arising from such processes is their complexity which makes it close to impossible to define a routine process as a guideline for professionals to use. This results in a process definition which cannot be used for similar problems concerning system architectures as they

almost certainly have special requirements not considered in a predefined solution.

From this problem that notations such as BPMN, which are executed before process execution, bring along, two requirements can be derived that are necessary to allow the modeling of processes that have a weak structure. On the one hand such processes should be highly flexible and need to allow each stakeholder to contribute without restricting other people working on the same process. On the other hand such processes need to provide ways for stakeholders to alter the process before run-time but also especially at run-time. As knowledge-intensive processes are constantly undergoing change, they should always allow for improvements and also for the structure to be changed. This is also the main difference to routine processes which do not require to be altered at any given time.

This need for a way to model knowledge-intensive processes in a very flexible way lead to a new management field which is often referred to in literature as Adaptive Case Management (ACM) or simply Case Management. The main goal of this new paradigm is to make processes adaptive in order to achieve a high amount of flexibility throughout the whole life-cycle of a process. Swenson defines such systems as:

> **Case Management Systems:** "Systems that are able to support decision making and data capture while providing the freedom for knowledge workers to apply their own understanding and subject matter expertise to respond to unique or changing circumstances within the business environment" [39].

Such processes can be considered the exact opposite of routine work. They are weakly-structured and as a result of that cannot be easily modeled using available standards such as BPMN. The notation has proven to be useful for processes that are predefined and are executed the same way many times, but it lacks the flexibility needed by knowledge-workers especially when an alteration of a process is required at run-time. Di Ciccio et al. share this opinion in their work on Knowledge-intensive Processes:

> **Knowledge-intensive Processes:** "Process management approaches are often based on the assumption that processes are characterized by repeated tasks, which are performed on the basis of a process model prescribing the

> execution flow in its entire- ness. This kind of structured work includes mainly production and administrative processes. However, the current maturity of process management methodologies has led to the application of process-oriented approaches in new challenging knowledge-intensive scenarios, such as health-care, emergency management, projects coordination, case management" [6].

Due to this lack of a common specification for modeling such cases, the Object Management Group (OMG) released the Case Management Model and Notation (CMMN), which is a common specification for describing knowledge-intensive processes and it aims to make them interchangeable throughout different applications based upon their language specification.

With the theoretical specification of a standard for modeling knowledge-intensive processes on the one side and huge technological improvements that benefit computer supported collaborative work (CSCW) on the other, the focus of this thesis is on the extraction of a subset of functionalities from CMMN and to integrate them into a collaborative application to support knowledge workers. The application is developed alongside and integrated into the Darwin application, which is currently being developed at the sebis (Software Engineering for Business Information Systems) chair at the Technical University of Munich. The application is supposed to serve as a basis for further evaluation of software supporting knowledge-workers.

1.1. Motivation

Drucker [7] argued in 1969 that one of the major management tasks will be to make knowledge work productive. Even though a lot of progress has been made since the release of Drucker's work, a constant attempt to improve knowledge work is still present nowadays. One of the biggest challenges is the support of knowledge workers using modern information technology as well as the right level of guidance. While a number of business process modeling languages have been released over the years, there seems to be a growing need for something more flexible and less restrictive in order to explicitly consider a knowledge workers environment.

The recent release of the CMMN definition in May, 2014 by the OMG moved the specification out of the beta stage. While this shows the relevance of modeling knowledge-intensive processes on the one hand it also shows that the research in

the field has gained some maturity and is now at a stage where specifications need to prove themselves in the real world on the other. By making use of notations such as CMMN it should now be possible to create full-scale business applications which rely on such concepts and ideally provide a way for exchanging information between different solutions of the same kind.

Case management, the foundation of CMMN, is the result of a continuous attempt to allow modern processes to be modeled, since contemporary workflow management systems do not provide the functionality and flexibility needed by knowledge-workers. Forrester [19] analyzed the following business trends as drivers which make case management so important:

- an increased need to manage the costs and risks of servicing customer requests
- the desire to automate and track inconsistent events which are weakly-structured
- a growing pressure on government agencies to handle more customer requests
- external regulations which require businesses to repsond accordingly
- the use of technology such as collaborative tools and social media to support business processes

Due to these developments this work focuses on CMMN from a practical point-of-view. One of the goals is to analyze to which extent CMMN can be applied to model knowledge-intensive processes in actual business cases and ultimately support their constant improvement and alteration.

The fact that this area of research is still at an early stage makes it especially interesting. Some of the most valuable papers cited in this thesis have just been published and many authors state that there is still a lot of gaps which need to be filled by future research. The motivation behind this thesis is to provide some insights into this area of research and to also fill some of these gaps. Marin et al. , who have recently analyzed the complexity of CMMN, state: "Another venue for future research is to identify subsets of the CMMN notation. As process modelers begin to use CMMN, it will be useful to identify the subsets of the specification that start to emerge [...]" [22]. This can be regarded as the main goal of this thesis and also the motivation behind it.

1.2. Problem Description

The attempt to define a common reference-model for the implementation of an application supporting knowledge-intensive processes resulted in the definition of a notation. It is up to the developers of applications which aim to tackle the hardships faced throughout the execution of such processes, to make use of the different elements and rules defined by a language like CMMN.

So far, a complex analysis of the applicability of the notation with regard to a large variety of processes originating from many different fields of work is hard to find. A lot of research has been focusing on different aspects of knowledge intensive processes. When the term case management emerged the main focus was on the improvement of contemporary workflow management systems as well as the process modeling languages, such as BPMN, which they were using. Van der Aalst et al. [44] introduced the name case handling in 2005, which can be regarded as one of the key events concerning the research on case management. Due to the complexity of BPMN 1.2 other experts in the field such as zur Muehlen created subsets of BPMN in order for it to be more use-case specific and easy to understand (cf. [49] [50] [51]). While this can be considered a good solution for making use of a notation already available, it should also be seen as a temporary one. The problem of the complexity of BPMN was rather concealed than solved. Fahland et al. [9], [10] support this view by confirming that maintainability and understandability are important when looking at knowledge-intensive processes.

The goal of the Case Management Model and Notation by the OMG is to solve this problem. The release of the specification is a direct result of the ongoing research in the area of case management. Researchers have become aware of the problems contemporary process modeling languages bring along when handling knowledge-intensive processes.

As a result of the release of CMMN a new problem arises. The new specification still needs to be adapted by process modelers on the hand and prove that it is capable of supporting different knowledge-intensive processes which are often unique in nature and require a lot of flexibility.

This work addresses this problem in particular. It looks at the different developments that have led to the release of a new specification. Furthermore, the disadvantages which many process modelers saw in contemporary languages such as BPMN are analyzed thoroughly, in order to assess whether CMMN is capable of handling these issues.

1.3. Research Scope

While the main focus of this thesis is on CMMN and its support of knowledge-intensive processes, a lot of related artifacts are analyzed if they contribute to the understanding of the topics discussed. This makes it important to set the research scope as a means of specifying what areas this thesis focuses on. Due to the constant research going on in this field, this work focuses on one area rather than attempting to fill all the gaps previously mentioned.

Generally, this work is focused on the extraction of a subset of CMMN and its implementation into an application focused on supporting knowledge-workers. In order to accomplish this, not only the CMMN specification needs to be considered, but also the previous research which ultimately led to the release of it. This is especially important as it helps to understand the gaps in process modeling that CMMN tries to fill.

The subset which is extracted from CMMN is based on the research performed over the course of this thesis. It should be considered a proposition rather than a complete solution as the application is still being developed. An empirical analysis concerning the usability of the implementation is not part of this work and can be part of future research.

1.4. Method and Outline

The main focus of research in this thesis is on collaborative knowledge work and the underlying processes. This requires this work to be structured in a way that makes it understandable. Due to the novelty of this topic, a thorough introduction is necessary to provide the required knowledge to understand the methods used and choices made throughout this work.

The following sections describe the scientific structure of this thesis. A definition of the methodical research approach is followed by a detailed analysis of the research objectives which this work aims to address.

Finally, the structure of the thesis is described by providing an overview of the chapters and topics addressed. This structure is based on three research questions which represent the logical outline and serve as a guideline for the research performed throughout this project. They are explained in detail below.

1.4.1. Methodical Research Approach

According to Hevner et al. [46] research regarding information systems can be classified into two different methodical approaches. On the one hand, the Behavioral-science paradigm focuses on observation. Its main objective is to describe the interaction of humans with a specific information system. On the other hand, the Design-science approach does not observe what is already available, but intends to develop new artifacts. Building on the available research for a certain field, those newly created artifacts should contribute to the solution of a specific problem in that area of research.

The fundamental scope of this thesis is defined by the Design-science approach. The developed application represents the artifact which was built considering the results of research in the area of adaptive case management. The fundamental problem in this case is the need to support knowledge-workers efficiently and purposefully, as well as the lack of available solutions that manage to solve this problem.

1.4.2. Research Objectives

The primary objective of the research performed in this thesis is to analyze knowledge-intensive processes thoroughly, and to extract a list of items that an application supporting knowledge-workers needs to provide. These items should represent a subset of those offered by CMMN. As a pre-condition, the Case Management Model and Notation needs to be analyzed and evaluated in order asses its applicability. Due to the fact that at the time of writing, the CMMN 1.0 specification has just been released and tools supporting it have not been published yet, this analysis is often based solely on the document provided by the OMG itself. Thus, it should only be regarded as an initial analysis which is not complete.

In order to create an exhaustive portrayal of the requirements of such an application, another objective is to factor in the research previously performed in the area. In many cases the authors described their own solutions, which makes it possible to analyze their key features and compare them to those extracted in this thesis.

It is also important to match the key elements extracted in this work against common workflow patterns. By doing so it is possible to evaluate the expressiveness of the application and to calculate its complexity.

1.4.3. Structure of the Thesis

This thesis is divided into three parts in a logical order. The first part focuses on the thorough explanation of the terms and specifications used throughout the course of the thesis. It also outlines the general structure and the methods used in order to explain the research topic and the corresponding objectives.

Followed by the introduction, the second part of this work focuses on the concept and addresses the research objective of finding a suitable subset of the CMMN specification.

Addressing the final research objective of finding a way to integrate the findings into an application based on the subset extracted from CMMN, the third part of this thesis is focused on the implementation itself.

The last part of the thesis is intended to outline the conclusions drawn from both the analytical as well as the implementational parts included. The focus is also on the proposition of areas of future research that succeed this thesis. This is especially important as the CMMN specification has just been released and needs to be analyzed thoroughly in order to evaluate its applicability in actual use-cases.

The following three research questions represent the comprehensive structure of this thesis based on the research objectives mentioned. They are analyzed chronologically and will be evaluated in detail to produce an elaborate answer that leads to a coherent theoretical and practical composition.

- **Research Question 1: How can CMMN support users with the definition of Knowledge-intensive Processes?**

  This question focuses on the analysis of the Case Management Model and notation as defined by the Object Management Group and its pertinence for the modeling of knowledge intensive processes. The different components of CMMN are introduced, explained and evaluated. This analysis will be based on the available literature on the one hand but also the use of actual examples of knowledge intensive processes that are modeled using CMMN on the other hand. In the end, this should result in a comprehensive evaluation of CMMN highlighting the advantages and disadvantages of the specification with regard to the broad range of fields in which knowledge intensive processes occur.

- **Research Question 2: What is a suitable subset of CMMN that can be used for Knowledge-intensive Processes?**

As the CMMN specification is very complex in nature, the main focus is on the extraction of a subset of elements included in CMMN that are suitable for the implementation into a collaborative application that supports knowledge intensive processes. This includes the evaluation of specific elements that emerge as suitable for applications, providing a detailed analysis of the advantages and disadvantages that they might implicate. If a specific element of the specification is found not to be suitable for an integration into an application, a detailed evaluation is used to support this decision. Throughout this process the complexity of such an application is always included in the evaluation, since the focus of the application in question should be less on completeness with regard to functionality as specified by CMMN, and more on usability by a wide range of knowledge workers that operate in different fields.

- **Research Question 3: What is a suitable software environment for CMMN?**

  While the previous research question mainly focuses on the theoretical analysis of the different elements included in CMMN with regard to an application supporting knowledge intensive processes, this final task represents the actual implementation of a run-time system that incorporates CMMN. In order to accomplish this, an actual research application which aims to support knowledge intensive processes will be extended to support the previously extracted set of CMMN elements. The main focus is on the analysis of the different ways in which the elements can be implemented and an explanation of the reasons for implementing a functionality in a certain way. The goal of this approach is to evaluate the actual applicability of the specifications included in CMMN to support knowledge workers that make use of a collaborative run-time system.

2. Theoretical Background

This chapter serves to give a detailed overview of the different terms used throughout this thesis. They play an important role in the following parts and thus should all be explained thoroughly. It starts with the definition of knowledge work, which is considered the basis of this area of research and explains what types of professions and fields can be attributed to knowledge workers. Subsequently, knowledge intensive processes which are executed by knowledge workers are outlined. In addition, two common terms to distinguish processes are analyzed as they resemble an important approach to categorize them. Finally, the Case Management Model and Notation is explained and analyzed in detail in order to lay the groundwork for a further evaluation.

2.1. Knowledge Work

> *"To make knowledge work productive will be the great management task of this century, just as to make manual work productive was the great management task of the last century."*
>
> *Peter F. Drucker, 1969* [7]

The term knowledge work has been used to describe the work done by workers with special knowledge. As outlined in the quote above by Peter Drucker it is considered to be one of the key success factors in modern companies. This is mainly due to an increasing number business processes that rely heavily on knowledge work. Routine processes can be executed, at least to some extent, by machines and can be easily modeled. Knowledge-intensive processes on the other hand are usually hard to model and cannot be turned into a routine. The following sections aim to explain this condition in detail, in order to lay the foundation for a more detailed analysis of the problems faced when modeling knowledge-intensive processes.

2.1.1. Knowledge

In order to explain the term Knowledge Work it is important to start with the actual definition of the word knowledge and how it differs from related terms such as information or data. Data can be considered as a set of characters that is following the rules of a predefined syntax (e.g. English language, a mathematical formula). The fact that the data is materialized in a certain form and with a syntax does not make it information yet. It is the process of putting the data at hand into a context, which creates actual information that is useful to a person. The most significant part is the creation of knowledge from the available information. This step involves the integration of the information, making it relevant to the person that processes the information. This relationship is illustrated in figure 2.1. The knowledge-creation process can be seen as linear and a specific action is the facilitator between two levels within the process.

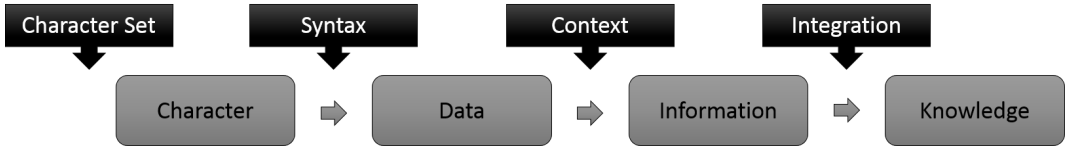


Figure 2.1.: How knowledge is created. Author's own compilation based on Rehuser, Krcmar 1996 [35]

Knowledge is strongly connected to a person and that person's ability to put information into context. A good example to explain this is a kid in primary school. It is able to read and process information that it considers relevant, but if that kid is shown a complex formula used in theoretical computer science, it will most likely not be able to make use of that information. This would lead to a processing of information without the creation of knowledge, as the contextualization is not possible in this case (i.e. the school-kid does not know anything about informatics).

Nonaka and Takeuchi, 1995 nonaka1995knowledge state that knowledge can only exist in the context of a person and that person's beliefs and experience. Davenport and Prusak support this view of knowledge by giving the following definition:

> **Knowledge:** "Knowledge is a fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for eval-

> uating and incorporating new experiences and information" [3].

The differentiation between tacit and explicit knowledge is used by Polanyi [33], in order to describe the ways in which knowledge can exist. While explicit knowledge is easy to grasp and can be documented and distributed using an appropriate way of codification, tacit knowledge resembles personal and context-specific knowledge which is hard to formalize. Both terms are not exclusive and explicit knowledge can be considered a part of tacit knowledge which can be externalized (cf. [25, 12]).

2.1.2. Knowledge Work Definition

With the detailed definition of knowledge in the previous section, it is now possible to define the scope of knowledge work and to demonstrate how it differs from other types of work. Knowledge Work is strongly connected to the tacit part of knowledge that was previously discussed, which cannot be easily extracted and documented. It should also be noted that the literature on Knowledge Work often states that it is sometimes difficult to define a certain area of work as exclusively knowledge intensive and there are often different opinions about what can be assigned to Knowledge Work.

One approach to better classify knowledge work, as proposed by Hube [15], is shown in Figure 2.2. There are two dimensions which best describe the amount of knowledge work in a specific process. On the one hand, the novelty of the required tasks to complete a unit of work measures how much that process differs from its predecessors. With an increase in novelty, the need for new approaches to complete the work rises. On the other hand, the complexity of the work can measure the level of expertise needed. A combination of work with a high degree of novelty and complexity is the area where most processes can be classified as knowledge work. It should be noted that knowledge work can be found in any combination of the two dimensions. Nevertheless, it is important to be aware of the key indicators that define the degree of knowledge work.

De Man at Cordys [4] uses a different approach by defining three categories to rank cases according to their need for special knowledge.

- **Mass cases** The traditional view of a process. It allows a workflow management system to fully automate and manage all activities executed.

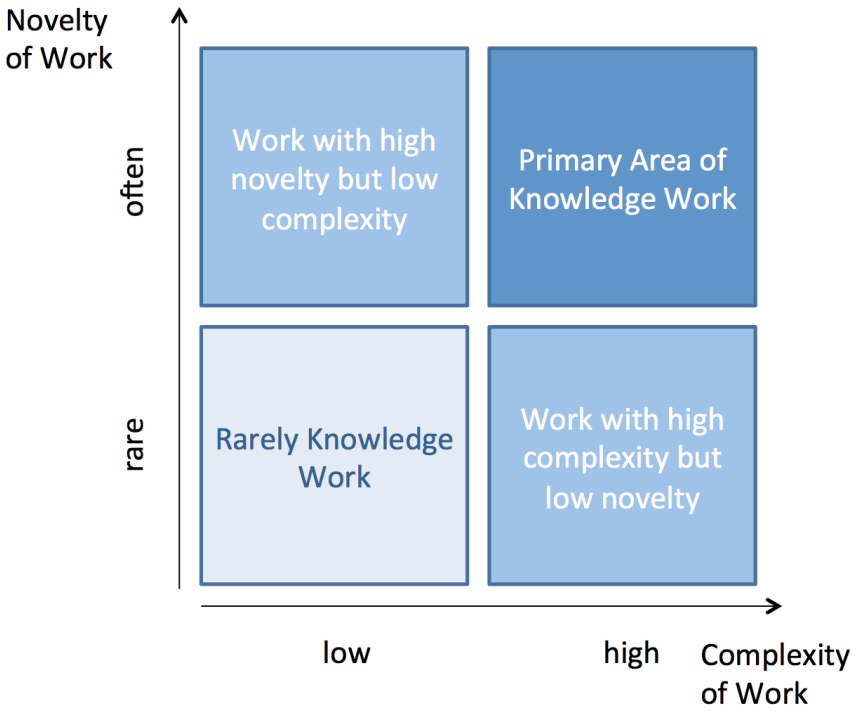


Figure 2.2.: Different areas of knowledge work. Author's own compilation based on Hube, 1995 [15]

- **Regular cases** A process that involves the knowledge of human workers, but which has certain constraints such as business rules that might restrict certain activities from being executed.

- **Special cases** These processes are defined by a high degree of freedom and an application does not restrict the user's actions but is used for support. A high degree of knowledge is required to execute these activities that are often unique during each iteration.

The term case is discussed more thoroughly throughout the next chapters and plays an important role in the theory behind knowledge intensive processes. While the categorization of different business processes may differ a lot in literature, the above classifications help understand the different types of processes that may exist within a company's environment. The latter two cases are both defined by their high degree of knowledge that is at least partially required.

2.1.3. Knowledge Workers

Now that a clear definition of knowledge work has been outlined, it is important to look at the people that are doing the actual knowledge work. In the literature they are commonly referred to as knowledge workers for which Davenport gives the following definition:

> **Knowledge Worker:** "Knowledge workers have high degrees of expertise, education, or experience, and the primary purpose of their jobs involves the creation, distribution, or application of knowledge" [3].

This definition of knowledge workers shows that knowledge workers are often found in areas where people have a high degree of qualification. This includes but is not limited to knowledge-intensive industries. A manager in basically any company can be considered a knowledge worker, as well as engineers and researchers in industrial companies [3]. The fact that virtually any person doing work that requires special knowledge can be considered a knowledge worker, makes it hard to set the limits of what is still knowledge work. It is also important to note that there is an increasing number of jobs that require special knowledge nowadays [3]. In order to still distinguish them, Drucker defines a knowledge worker as "someone who knows more about his or her job than anyone else in the organization" [8].

Another important aspect to consider is a knowledge workers way of doing work. As there is often a lot of expert knowledge involved, a typical knowledge worker might consider restrictions in a process as obstacles rather than assistance. As workflow management systems, which are discussed in the following chapters, rely heavily on such restrictions to model a business process, a knowledge workers attitude towards such a system might be influenced a lot depending on the consideration of knowledge work within a contemporary process modeling application.

2.1.4. Knowledge-intensive Processes

As this thesis is especially focused on knowledge-intensive processes it is also necessary to specify the characteristics of it. This is particularly important, because not every task executed by a knowledge worker has to be also knowledge-intensive.

An important indicator of the importance of handling knowledge-intensive processes adequately is the development of knowledge work over the course of time. Figure 2.3 shows this development. In the past, most processes could not be attributed to knowledge work as they mainly contained physical tasks. Nowadays this condition has changed to the opposite with machines doing most of the physical activities, while human workers can focus on tasks which require their expert knowledge.

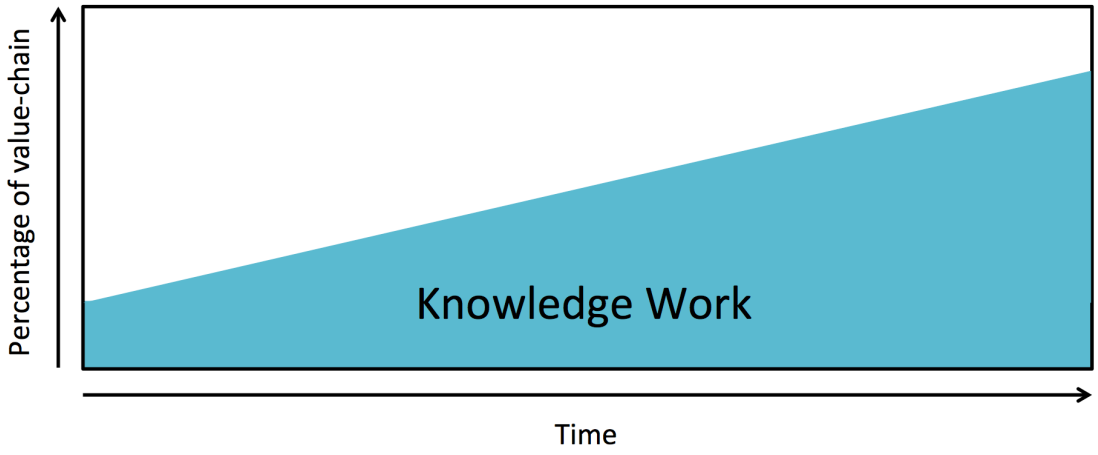


Figure 2.3.: The role of knowledge work in the economy. Author's own compilation based on Pfiffner and Stadelmann 2012 [32]

The general understanding of a knowledge-intensive process in this thesis is a process which includes activities that require human expertise in order to be completed sufficiently. Furthermore, these activities are often unique in nature and usually require a different approach during each iteration. Such processes rely on human input in a special way, making it hard to predefine a set of activities that reliably lead to their completion. Thus, it is also not possible to generate a control-flow which is used by contemporary workflow management systems.

Ciccio et al. give the following definition for Knowledge-intensive Processes:

> **Knowledge-intensive Processes 1:** "Processes are defined knowledge-intensive when people/agents carry them out in a fair degree of "uncertainty", where the uncertainty depends on different factors, such as the high number of tasks to be represented, their unpredictable nature, or their dependency

> on the scenario. In the worst case, there is no pre-defined view of the knowledge-intensive process, and tasks are mainly discovered as the process unfolds" [5].

This definition shows that the main attribute of knowledge-intensive processes is their uniqueness. In [6] they mention that their understanding of Knowledge-intensive Processes is best described by the definition given by Vaculín et al. which focuses on the data-centricity:

> **Knowledge-intensive Processes 2:** "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 run-time" [42].

In this more recent work, the authors also extract a set of characteristics of Knowledge-intensive Processes [6]:

- **Knowledge-driven** Data and knowledge influence the process and human decision making.
- **Collaboration oriented** Processes usually involve a number of people that work together. Process participants usually have different roles.
- **Unpredictable** Activities within a process can change or be replaced at any given time. This can be the case at design-time, during execution or among different instances of a process.
- **Emergent** These types of processes cannot be defined beforehand and usually emerge over time as more and more information becomes available.
- **Goal-oriented** Instead of focusing on the execution of specific activities, the process is defined by goals that lead to the completion of it.
- **Event-driven** Different events occurring during the run-time of the process define the nature of its execution.

- **Constraint- and rule-driven** The processes may have some rules that define the way in which they can be executed.
- **Non-repeatable** Single instances of such processes usually differ from other instances in a way making it hard to predefine a control-flow.

While they can have attributes that show certain patterns it is much more likely that parts of the process are entirely unique to an iteration. A control-flow as it is used in contemporary workflow management solutions usually doesn't leave room for maintainability. For that reason, a lot of research has been conducted by experts in the area of process modeling, in order to come up with methods that are specifically designed to handle knowledge-intensive processes. This issue will be addresses in the following parts of this work.

2.2. Declarative Processes

The goal of this section is to show why declarative processes play an important role when trying to model knowledge-driven environments. In order to understand what declarative processes are, it is important to first look at their counterpart: imperative processes. Examples will be taken from software development as similar concepts also exist in programming. Fahland et al. [9] support this approach as they also see many similarities and did not encounter any strong counter arguments.

In programming, imperative programming styles are concerned with specifically how an application or method is executed. O'Regan provides the following description:

> **Imperative Programming:** "Imperative programming is a programming style that describes computation in terms of a program state and statements that change the program state. [...] Similarly, imperative programming consists of a set of commands to be executed on the computer and is therefore concerned with how the program will be executed. The execution of an imperative command generally results in a change of state" [27].

The term imperative implies that the programmer is aware of the underlying logic to achieve a certain task, providing the specific functions on his own. This

stands in contrast to the declarative programming style which rather "involves stating what is to be computed, but not necessarily how it is to be computed" [20].

Both terms can be compared to the two types of processes discussed in the previous chapters. The imperative programming paradigm can be compared to Business Process Modeling (BPM), as the modeled processes are also specified in an imperative manner. The different stages within such a process usually come with a predefined order and contain a lot of information on how to perform its different task to reach the specified goal. Declarative programming on the other hand has strong similarities to knowledge-intensive processes in terms of goal specification. As such processes are executed by experts, they predominantly contain less information on how to achieve a certain task within the processes, but rather leave it up to the knowledge worker to decide how to tackle a specific problem. It is the final objective of the process that defines the actions taken by the people involved.

2.2.1. Imperative Process Models

Traditionally business processes were modeled using a strictly imperative approach. Workflow management systems assist a user by providing data derived directly from an underlying control-flow. The user of such an application is usually not able to make local decision and needs to follow the strict order of tasks. These types of process models can be categorized as imperative due to their explicit nature.

Fahland et al. [9] argue that an imperative process model is most suitable for repetitive processes that contain very little circumstantial information. "Given two semantically equivalent process models, establishing sequential information will be easier on the basis of a model that is created with the process modeling language that is relatively more imperative in nature" [9].

2.2.2. Declarative Process Models

Pesic [30] argues that business processes have two opposing properties. On the one side flexibility is seen as the possibility for users (the people who execute such processes) to make ad-hoc "local" decisions during execution. Such decisions are random and do not necessarily follow a certain pattern. On the other side support is the enforcement of centralized decisions which a system uses to guide a user and to create a set of predefined boundaries.

According to Pesic, the two extreme types of business process management (BPM) systems are groupware and workflow management systems. While groupware tools offer high flexibility they usually lack the support that is sometimes necessary to efficiently work on certain problems that are still highly unstructured. Workflow management systems on the other hand usually expose users to problems that come with a large amount of predefined rules and constraints, making it hard to individualize ones approach.

As a means of finding an optimal way to support knowledge intensive processes, the challenge is to find a way to combine the two opposing sides in a way that is suitable to solve highly unstructured problems. As a possible solution, Pesic proposes the approach of declarative process models in which the main focus is on the overall objective rather than the specific subtasks that are part of the process. This is in contrast to the traditional control-flow model of workflow management systems, which define the order of tasks making the users stick to the order "explicitly specified in the control-flow" [30].

The need for a declarative process model becomes obvious, when looking at todays knowledge-intensive business processes which often have a high level of unpredictability. This view is also supported by Fahland et al. who propose that "establishing circumstantial information will be easier on the basis of the model that is created with the process modeling language that is relatively more declarative in nature" [9]. In many cases, it is not the control-flow that is previously available to the user but a certain set of constraints that sets the boundaries of the process. Pesic categorizes the activities executed throughout the life-cycle of a process into three distinct groups:

- **Forbidden scenarios**

  Even with the process being highly unstructured and sometimes unpredictable, there are scenarios which can still be explicitly excluded from the model. Those scenarios are outside the boundaries of the business process and are never considered throughout the execution. The forbidden scenarios resemble the constraints of a business process in the constraint-based process models as opposed to the traditional model which predefines what is possible.

- **Optional scenarios**

  Optional scenarios are not part of the core process, but can be applied if necessary. As they are optional scenarios, they are part of the process and lie

within the boundaries of the constraint-based approach which is in contrast to the traditional approach..

- **Allowed scenarios**

  The allowed scenarios represent the core features of a process, and can be executed whenever necessary. There is no explicit order, leaving the actual control-flow up to the user, which creates a high level of flexibility. Furthermore it is not necessary for a user to go through all allowed scenarios. Instead a substantial subset of those scenarios represents the anticipated outcome of a process execution.

Figure 2.4 shows the difference between the traditional approach and the declarative one. The traditional, imperative process contains a predefined control-flow that gives the user a lot of support, while restricting the actions to the boundaries defined at design-time.

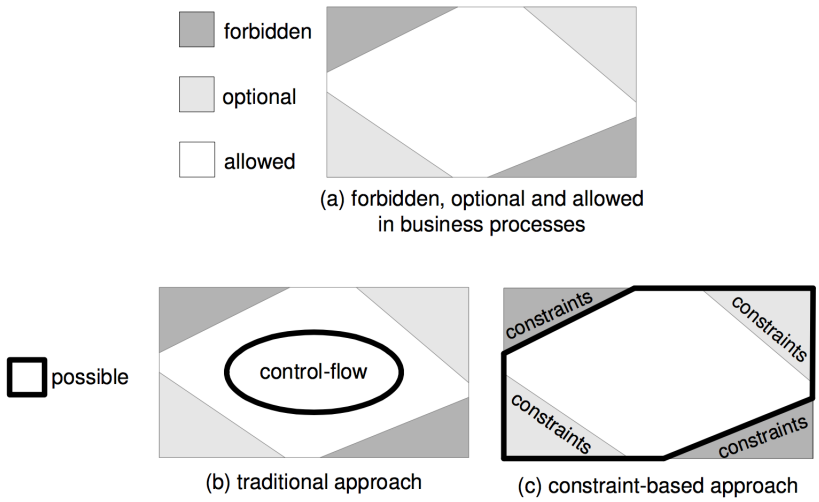


Figure 2.4.: Imperative and declarative approach [29].

Applying the three scenarios explained, the declarative or constraint-based approach uses these different types of scenarios to define more natural boundaries. While the forbidden ones, which were extracted at design time, cannot be executed by the user, there are no other artificial restrictions. The users are free to execute tasks at their discretion, which offers a lot of flexibility.

2.3. CMMN

With the intention "to supplement the procedural perspective of BPMN" [22] the Object Management Group (OMG) has released the Case Management Model and Notation (CMMN) 1.0 specification in May, 2014 five years after the initial request for proposal (RFP) in 2009. CMMN differs from other business process modelling notations due to its focus on data. Furthermore, the specification incorporates the aforementioned declarative process modelling approach. This chapter focuses on the detailed analysis of the specification, evaluating some of the key elements and features. It also explains the problems users of contemporary modeling languages faced and how case management attempts to solve them.

2.3.1. Case Management

Case management or case handling was introduced due to the fact that experts in the field criticized the restrictive nature of contemporary workflow management systems. According to van der Aalst et al. [44] this results in a lack of flexibility and consequently also usability. The authors highlight four problems that arise as a result of this restrictiveness:

- **Atomic activities**

  Since a workflow management system considers activities performed by a user to be atomic, there is no possibility to handle activities that aren't. Sometimes activities are handled by users in a much more detailed and complex way, but due to the requirements of workflow management systems need to be turned into atomic ones.

- **Routing for distribution and authorization**

  Generally, contemporary workflow management systems distribute work according to the level of authorization of its users. While this approach is useful for distributing activities only to users that have the right privilege it lacks a strategy to distribute work using different logic. For example, a person with a high authorization level does not necessarily need to see all the work he is allowed to view.

- **Context tunneling**

The context of the actual business case handled by a workflow is not at the center of attention since the focus is on the underlying activities.

- **Implicitness**

  Activities within the control flow are considered essential to the process. For that reason, users are forced to complete those activities in order to complete the workflow. This results in a decrease in flexibility.

As a result van der Aalst et al. "propose case handling as a new paradigm for supporting knowledge-intensive business processes" [44]. Addressing the problems mentioned, the authors come up with the following core features of case handling:

- **Context tunneling prevention**

  Preventing the user from losing focus due to an inappropriate integration of the actual case handled.

- **Using available information**

  Determining the order of execution by using the information available instead of just following a pre-specified order of activities.

- **Roles**

  Making use of multiple roles such that it is possible to efficiently and appropriately distribute information to the right resources.

- **Always allow process alteration**

  Allowing users to add and change information at any given time, to allow for more flexibility.

The result of this problem analysis is the proposition of a new case handling paradigm that consists of a *case* as the central component. This case contains a number of *activities* that users can execute. A major difference to contemporary workflow management systems is the non-atomic way in which those activities are specified. This was one of the major problems quoted above. The *process* is the representation of the connections between such activities. The authors discourage the use of too many precedence relations as they take away much of the flexibility a knowledge worker expects. Furthermore the authors state thate a knowledge-intensive process is "based on a collection of *data objects*"[44].

Another integral part of case management is the supporting use of *roles*. Van der Aalst. et al. refer to these roles as actors that have certain abilities at their disposal:

- Execute role
- Redo role
- Skip role

The roles can be set for each activity according to its specific requirements [44].

The main differences between contemporary workflow management systems and the case handling paradigm as proposed by [44] are outlined in Table 2.1.

	Workflow management	**Case handling**
Focus	Work-item	Whole case
Primary driver	Control flow	Case data
Separation of case data and distribution	Yes	No
Separation of authorization and distribution	No	Yes
Types of roles associated with tasks	Execute	Execute, Skip, Redo

Table 2.1.: The differences between workflow management and case handling. Author's own compilation based on van der Aalst et al., 2005 [44]

2.3.2. Origin of the Specification

The need for a new specification to meet the demands of modern business processes has been growing recently. As these processes often have certain attributes such as a weak structure and a high grade of uniqueness, traditional languages do not offer the required flexibility. For that reason, the OMG filed a request for proposal in 2009 to define a standard [22]. Experts in the field have reinforced the need for a different modeling approach for knowledge-intensive business processes years before the RFP and the official release of CMMN. For example, the work of van der Aalst et al. [44] on the aforementioned case handling deals with the problems of most contemporary workflow management systems.

Another important contribution that does not directly refer to case management was the introduction of Business Artefacts at IBM Systems by Nigam and Caswell [24] in 2003. These business artifacts as opposed to business objects "model the lifecycle aspect" [21] which supports the view of a knowledge-intensive process as a case. Marin et al. [21] also argue that the Adaptive Documents (ADocs) introduced by Kumaran et al. [18] show many similarities to Business Artifacts. Even though these concepts are not always explicitly mentioning case management, they illustrate the growing need for a new paradigm throughout the past decade.

According to Marin et al. [21] another important development towards the introduction of CMMN was the shift from a strictly procedural to a more declarative lifecycle model. They refer to Vortex [16] which was introduced in 1999 as being the first data-centric framework which "supports highly flexible workflows" [21]. Because of the specific development of Vortex for "personalization applications in call routing and web store fronts" [21], the introduction of the guard-stage-milestone (GSM) approach can be viewed as a generalized specification of the features included in Vortex, which contains less restrictions. Due to the active participation of the creators of the aforementioned specifications, such as IBM and Cordys, many features have found their way into the final version of CMMN. An example for this is the behavioral model of GSM which is used in CMMN as well (cf. [21]).

2.3.3. Target Users

Just like other business process modeling languages, CMMN is intended for professional users. While knowledge workers without specific expertise can create Case models on their own, the actual task of extracting an optimized version using multiple cases is intended for advanced users. The official CMMN 1.0 document states that "business analysts are the anticipated users of Case management tools for capturing and formalizing repeatable patterns of common Tasks, EventListeners, and Milestones into a Case model. A new Case model may be defined as entirely at the discretion of human participants initially, but it should be expected to evolve as repeatable patterns and best practices emerge" [26].

This shows the presence of two different groups of users. The case workers on the one hand, execute a process and keep adding information by creating information items in CMMN whenever they need to. The professional case modelers on the other hand try to make use of the various versions created by the case workers

continuously improve the process by extracting patterns.

2.3.4. Structure of the Notation

This section focuses on the introduction of the structure of CMMN. Some of the specifications most important meta-models are used to explain the theory behind the notation. While looking at the outermost structure of CMMN on the one hand, the most important underlying components are analyzed thoroughly on the other.

Core and Case Model Elements

As shown in Figure 2.5, the *Definitions* class is the containing object of all elements, while Definitions inherits from *CMMNElement*, making each object in CMMN related to it. Generally, the definition of an object can be regarded as its basic information containing things such as namespace, creation date, and author.

The *Import* class is used for referencing external type definitions. This makes it possible for the *CaseFileItemDefinition* to reference these external elements. The document of the specification does not provide a list for supported types but uses XSD as an example, indicating that it should be the most commonly used.

A *Case* is the class that represents its equivalent in Case management. While it contains information on its associated roles and defines the case's name, it also has optional input and output *Parameters* which enable other cases to make use of the information produced by it. This can be compared to the input and return parameters of methods in a programming language.

Information Model Elements

Each Case contains exactly one *CaseFile* and one *casePlanModel*, which is explained below. *Stages*, which will be analyzed further in the next chapter, can be regarded as container elements which help structuring a *Case*. The outermost *Stage* of a *Case* is defined as its *casePlanModel*.

Roles are important feature of CMMN. As already mentioned by van der Aalst et al. [44], they are necessary to provide context specific information for the right resources efficiently. In the specification they are designed to authorize case workers or a group of them to execute *HumanTasks* and to raise user events.

The document of the specification uses Doctor, Patient and Nurse as example roles to illustrate the different types of authority and case specific knowledge.

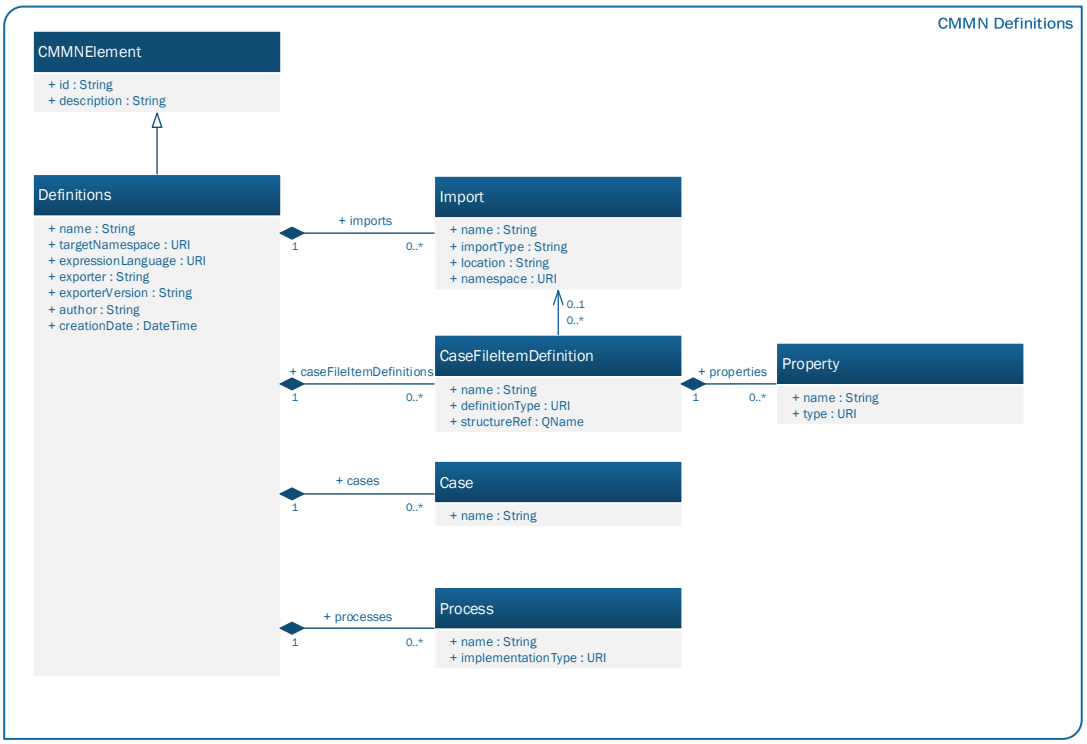


Figure 2.5.: The main class diagram of the CMMN specification. Author's own compilation based on [26].

In CMMN the information model is an essential element of the specification, which is mainly due to its data-centric nature. It contains all the classes required to manage information, or data, that is part of a *Case*. Its main elements can be seen in Figure 2.7.

A *CaseFileItemDefinition*'s relation to a *CaseFileItem* is comparable to the relation of all elements to the global *Definitions* class. Each *Case* consists of exactly one *CaseFile* which contains all the information added to that case. The *CaseFile* being a single element, can contain many *CaseFileItems*. The document of the specification gives the following definition for a *CaseFileItem*:

> **CaseFileItem:** "A *CaseFileItem* may represent a piece of information of any nature, ranging from unstructured to structured, and from simple to complex, which information can be defined based on any information modeling

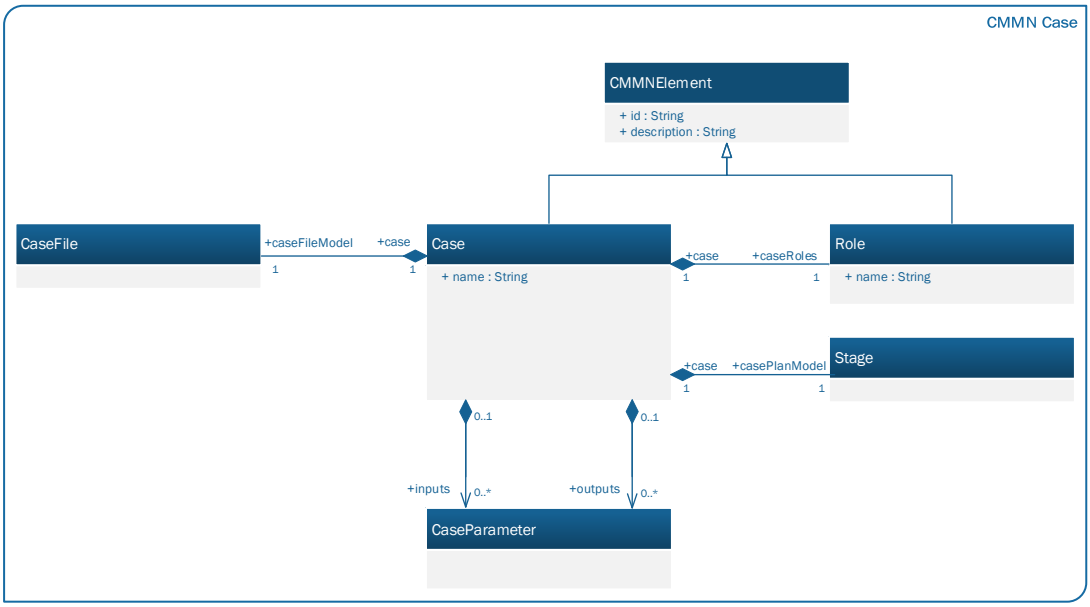


Figure 2.6.: The relations of the Case class in CMMN. Author's own compilation based on [26].

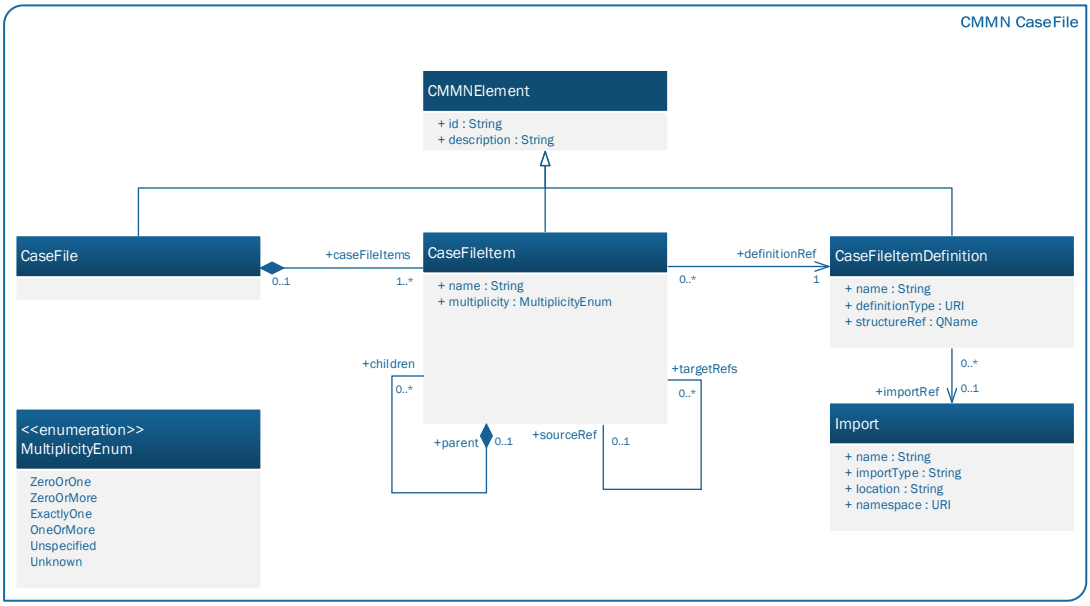


Figure 2.7.: The elements of the information model in CMMN. Author's own compilation based on [26].

> *language*. A *CaseFileItem* can be anything from a folder or document stored in CMIS, an entire folder hierarchy referring or containing other *CaseFileItems*, or simply an XML document with a given structure. The structure, as well as the *language* (or format) to define the structure, is defined by the associated *CaseFileItemDefinition*" [26].

It is part of CMMN's flexible nature to not specify the format of a file further. For that reason a *CaseFile* can be compared to a folder on a file system, which can contain virtually any format as files.

Plan Model Elements

The *casePlanModel* previously mentioned contains the elements of both, the initial structure of the case as well as those created throughout its continuous adaption during run-time. As already mentioned, a *casePlanModel* is regarded as the outermost *Stage* which "represents a recursive concept" [26].

The *PlanItemDefinition* is an abstract class as depicted in Figure 2.8. It is used to construct *Case* plans. It contains some of CMMNs core elements such as *EventListeners*, *Milestones*, *Tasks* and *Stages*. The *PlanItemControl* shown in Figure 2.8 is used to specify control data.

The *EventListener* class as shown in Figure 2.8 handles the events which occur during the run-time of a *Case*. Such events can be the changing of the state of a *Task* or *Stage* as well as the completion of a *Milestone.*

In CMMN there are natural "standard events" (cf. [26]) which can be activities such as the alteration of information within the *CaseFile*. These "standard events" represent transitions in the lifecycle defined by CMMN and are handled by *Sentries*. The *EventListener* class is intended to handle those events that are not within the boundaries of a *Sentry*.

The *EventListener* class has two subclasses which are used to differentiate between two types of events:

- *UserEventListener*: The UserEventListener catches events that are triggered by the users working on the Case.
- *TimerEventListener*: The TimerEventListener catches events that are triggered according to a previously defined timer.

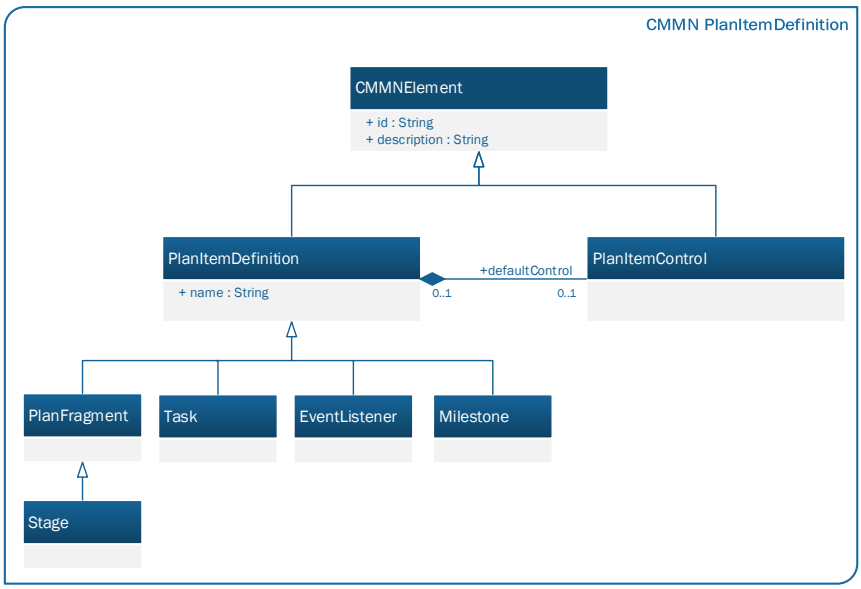


Figure 2.8.: Plan items in CMMN [26].

The *Milestone* class represents a target, which can be achieved by a section of the *Case*. It is used to calculate the progress of a *Case* at run-time. Next to the trivial connection of a Milestone with the completion of multiple *Tasks*, it is also possible to connect a *Milestone* with information contained in the *CaseFile*. As soon as this deliverable is available, the Milestone is marked as achieved.

A *PlanItem* refers to a *PlanItemDefinition* and is the result of the extraction of patterns. This is usually the case when a best-practice has been discovered in a set of process iterations. *PlanItems* can be part of *PlanFragments* which represent patterns such as a sequence of two *PlanItems*. A *Sentry* indicates a possible dependency of two *PlanItems* within a *PlanFragment*.

Sentries handle various combinations of events and conditions. In CMMN an event is handled by an *OnPart* while a condition is handled by an *IfPart*. The following three scenarios are possible:

- If an event occurs, a certain condition is evaluated. If the condition evaluates to *true* the action of the *Sentry* is executed.
- An event occurs which enables a *Sentry*. No condition is necessary.
- A condition evaluates to *true* enabling the *Sentry*. No event is necessary.

A *Sentry* always refers to a *PlanItem*. It can be either at the entry or exit point of a *PlanItem*. This will result in a *Task* or *Stage* being enabled or flagged as complete respectively.

The two classes that case workers will use primarily to structure and add information are *Stages* and *Tasks*. As mentioned before, a *Stage* is used to order and group items while *Tasks* represent single "atomic" units of work. Both are explained in detail below.

2.3.5. Visual Elements of the Notation

As mentioned in the previous sections, a big part of the CMMN specification has its roots in the literature and the resulting technologies discussed. One of the most important features is the "clear separation in CMMN between the case folder (information model) and the case behavioral model (lifecycle)" [21]. This section is focused on the analysis of the most important visual elements within the CMMN specification as defined by the Object Management Group. It is important to note that some of the elements in CMMN are discussed in detail while others are intentionally left out or have been described in detail in the previous chapter. The official document by the OMG [26] contains a detailed list of every available element of the specification.

Visual Components

As per CMMN specification, only the behavioral model is depicted using model elements. The information model is only visible if it is directly connected to the behavior of a case (i.e. CaseFileItem).

The main element in CMMN which specifies the process, which is being handled is a *Case*. The OMG defines a case as "a proceeding that involves actions taken regarding a subject in a particular situation to achieve a desired outcome" [26]. An important attribute of a case is its independent nature, making any iteration over it potentially unique.

In CMMN, a case is modeled using the *CasePlanModel* shape as shown in Figure 2.9, which resembles a folder specifying the boundaries of a case. As mentioned in the previous section, the CasePlanModel is "the outermost Stage that can be defined for a Case" [26].

With the *CasePlanModel* being the outermost element in CMMN, the *Stage*, which

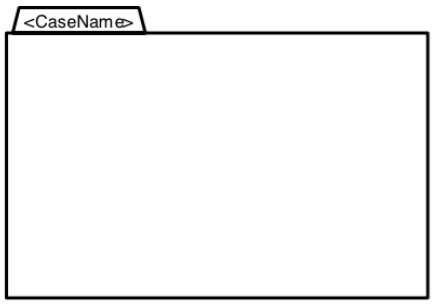


Figure 2.9.: The CasePlanModel shape [26].

is shown in Figure 2.10, is another important element, which is primarily used to group activities of a process. The specification also defines an option to collapse and expand a stage indicated by a ”-” and ”+” icon respectively.

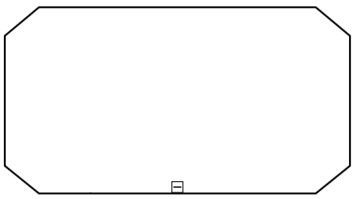


Figure 2.10.: A stage in its expanded state [26].

A *Task* in CMMN is defined as ”an atomic unit of work” [26]. While a stage can contain many tasks, a task is seen as an atomic activity that does not require a further division. Due to the amount of possible tasks, different types exist in order to specify the content of a task. *Human tasks* define tasks that are executed by a case worker. An example of a human task is depicted in Figure 2.11.

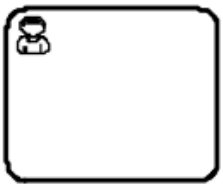


Figure 2.11.: An example of a human task [26].

Furthermore, tasks can also be *case tasks*, which are a reference to another case, and *process tasks*, which represent a reference to another business *process*. In CMMN,

a process is an abstract representation of a model from another language such as BPMN, XPDL or BPEL.

Stages and tasks can be both plan items and discretionary items according to the CMMN specification. While plan items are considered items that are already known during the design-phase, discretionary items can be executed at the case workers discretion. Thus, plan items can be considered the result of an analysis of best-practices, while discretionary items are left open for the case worker to decide if they are necessary. A dashed borderline is used in CMMN to indicate that an element is discretionary.

In order to visualize dependencies between two stages or tasks, CMMN uses *connectors* which are represented by dashed lines as shown in Figure 2.12. The white diamond shaped element is a *sentry* which specifies that the task has a dependency and is cannot be completed. In the case of Figure 2.12 Task B depends on Task A and is waiting for its completion.

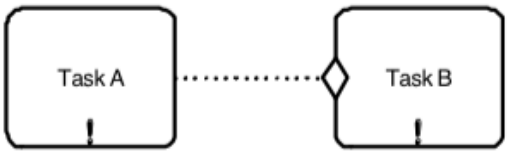


Figure 2.12.: Dependency between two tasks using connector and sentry [26].

Making use of connectors and sentries, it is possible to model common dependencies such as AND (see Figure 2.13) and OR (see Figure 2.16). The exclamation mark, which can be seen on the tasks represents a CMMN decorator marking a task or stage as required. There exist various decorators in CMMN, which are described in detail within the specification.

Other important elements included in the CMMN specification are *event listeners*, which wait for a timer or user event to occur, and *milestones* which have a specified number of entry criteria which indicate dependencies that need to be completed in order to finish them. As already mentioned in the previous section, either a timer or a human event can trigger *event listeners*. Figure 2.14 shows the visual representation of the the two types of *EventListeners*.

Milestones indicate the target of a section within the process. In Figure the connection of a milestone with a *Task* is shown.

The elements introduced represent the majority of items contained in CMMN. Their goal is to enable modelers to visualize cases of many different kinds. It is

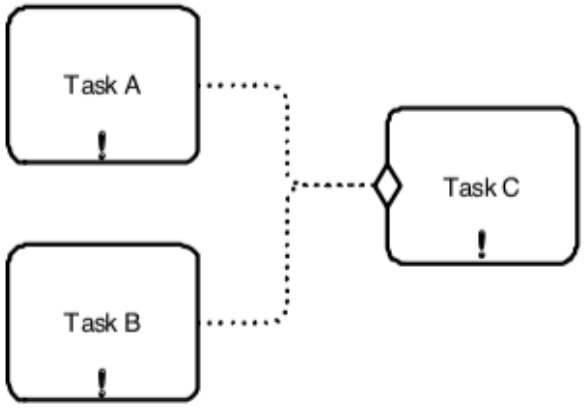


Figure 2.13.: AND dependency using connectors [26].

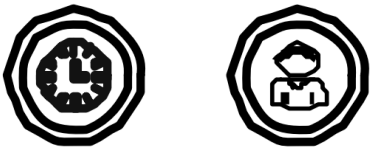


Figure 2.14.: Visual components to show presence of a *TimerEventListener* and *UserEventListener* [26].

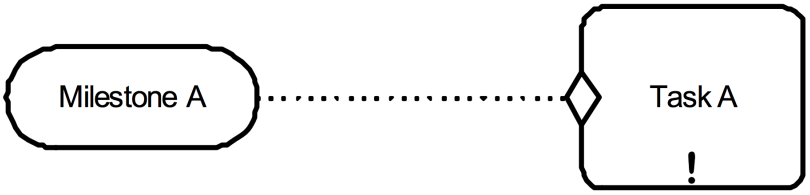


Figure 2.15.: A milestone connected to a Task. [26].

important to have an overview of the different elements contained in the CMMN specification to understand the concepts discussed throughout the course of this work.

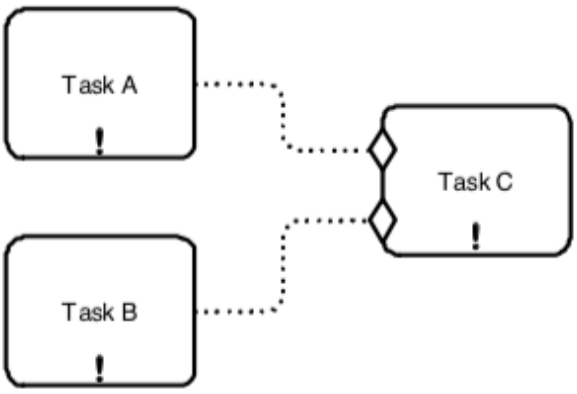


Figure 2.16.: OR dependency using connectors [26].

Use Case Example

An example of an actual case modeled in CMMN is the claims management example shown in Figure 2.17. It contains the majority of CMMN elements introduced and illustrates how they can be used to model a knowledge-intensive process. This section looks at this example in detail to explain the key elements in detail.

The *Claims File* represents the casePlanModel as well as the outermost Stage of the Case. The fact that it is used as a file rather than a process shows the focus on case management. With regard to the control-flow it can also be observed that the focus is not on the sequential ordering of the different activities like in other modeling languages, but rather on the input of data and the triggering of related events. The use of the casePlanModel as a Stage allows modelers to directly connect elements such as events and milestones to it. In Figure 2.17 this can be observed looking at the UserEventListener and Milestone (”Claims Processed”), which are directly connected to the casePlanModel.

The example also shows the use of Stages for the grouping of Tasks that belong to the same set of activities. The use of plan and discretionary tasks shows how it is possible to leave certain activities for the case worker to decide. The ProcessTask *Request Missing Documents* for example is only necessary if there are documents missing. By making that task discretionary it does not stop a case worker from leaving it open but is available as soon as it is required.

Process tasks such as *Identify Responsibilities* indicate that its execution leads to another workflow like BPMN. This gives CMMN the ability to reference a complex process using the expressive power of other modeling languages whenever suitable.

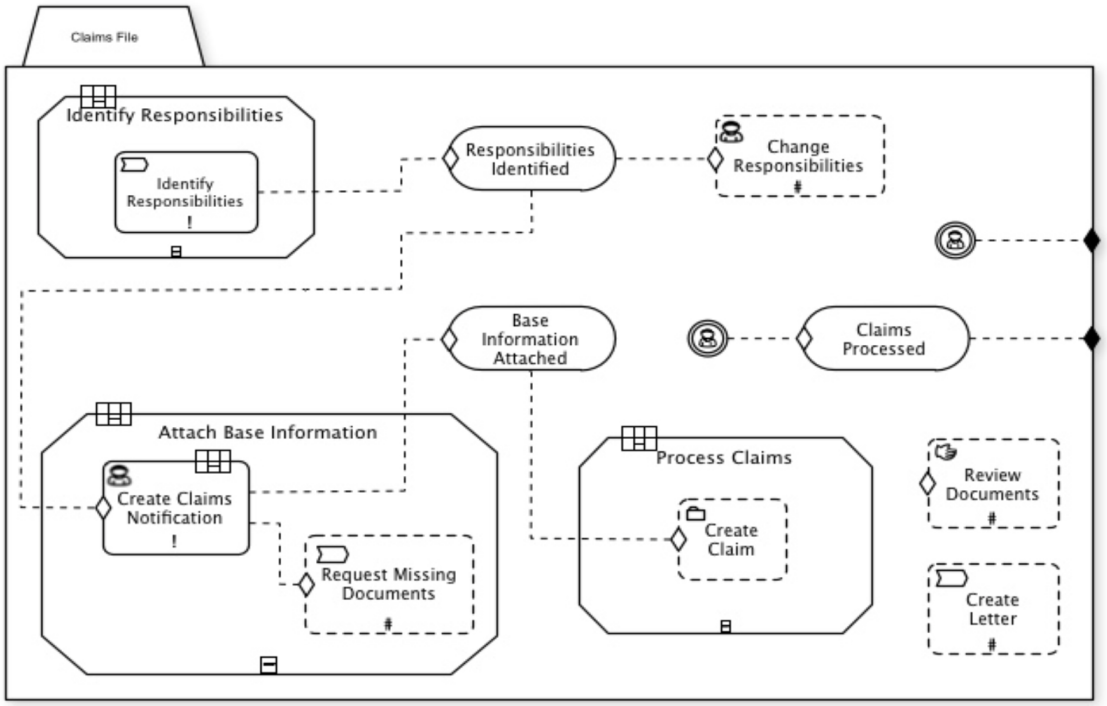


Figure 2.17.: Claims management example from the CMMN specification using most CMMN elements introduced [26].

Similarly, the task *Create Claim* is a CaseTask, which refers to another CMMN case. This is useful when the execution of a task has been modeled in CMMN as well.

Another aspect, which can be observed is the fact that connectors in CMMN can be used to create dependencies between many different elements. Good examples for this are the connection of a HumanTask with the *Claims Processed* milestone and the connection of the *Base Information Attached* milestone with the *Create Claim* task. This enables case modelers to create complex relations between a variety of CMMN elements.

3. Related Work

This chapter picks up the terms introduced in the previous chapter and focuses on the analysis of the scientific work related to them. Due to the recency of the topic, this is particularly important as the analysis serves as a basis for the approach used in the following chapters of this thesis. The goal is to create a timeline of the events that ultimately resulted in the release of CMMN 1.0. This also includes events that have indirectly contributed to the research on case management.

3.1. Process Models Supporting Knowledge Workers

Throughout the last two decades there has been a lot of research on process modeling languages that meet the requirements modern knowledge workers have. Many researchers have come up with new methods to model processes with a weak structure. This section combines some of the most notable work that has led to the invention of case management and ultimately the release of CMMN.

One of the main issues researchers had with the approach of contemporary process modeling languages such as BPMN was their attempt provide a notation that is capable of modeling even the most complex of structures within buiness processes. This can be regarded as one of the main reasons for experts in the area to extract simplified versions of BPMN 1.2 (cf. [41], [49], [50]). Theses simplified versions, which were basically subsets of the more complex parent, needed to exist in order to allow modelers to actively use them in their area of expertise.

Due to this key problem with contemporary modeling languages, Fahland et. al [9] raised the issue of understandability. They argue that there has not been enough research on the understandability of process modeling languages, which leads to new specifications being released in order to address certain issues not properly handled in a similar language. They conclude with a set of two propositions:

- **Proposition 1.** Given two semantically equivalent process models, establishing sequential information will be easier on the basis of the model that is

created with the process modeling language that is relatively more imperative in nature.

- **Proposition 2.** Given two process models, establishing circumstantial information will be easier on the basis of the model that is created with the process modeling language that is relatively more declarative in nature. Establishing circumstantial information will be easier on the basis of a declarative process model than with an imperative process model. [9]

Given these two propositions, they can be regarded as an indicator for the suitability of a process modeling language. Using the definition of knowledge-intensive processes from the previous chapter, their information can be regarded as mainly circumstantial. According to Proposition 2 by Fahland et al. this indicates that a process model best suited for knowledge-workers should be a declarative one. This issue is the focus of the discussion in the next section. While the propositions by Fahland et al. should only be regarded as an initial analysis of the issue at hand, they strongly support the opinion of many other researchers addressing this problem.

3.2. Imperative vs. Declarative Process Models

An important aspect of the research focused on case management is the shift from traditional process modeling which was mostly of an imperative nature towards a more declarative approach. Most researchers focusing on case management and knowledge-work refer to more declarative approaches, as they seem more suitable for their attributes.

Pesic and van der Aalst for example have focused their research on finding better approaches to handle knowledge-intensive processes (cf. [30], [29]). Figure 3.1 shows the major problem when finding the best way to support knowledge-workers with their processes. The diagram shows the issue of finding an optimal trade-off between flexibility on the one side and support on the other. Both sides can be seen as extremes that influence the behavior of software systems according to their paradigm.

Pesic [29] puts two different types of business process support systems on each side of the area of conflict by contrasting groupware with workflow management systems. The perception of groupware in his approach describes it as a software

system, which facilitates a lot of flexibility. Due to the very soft nature of such systems, their focus is on giving the user the freedom of choice instead of guiding them by using restrictions. In contrast to such non-restrictive systems, workflow management systems are depicted as strict due to their excessive support of users.

The two opposing sides, according to Pesic, either support centralized or local decision making. Centralized being the approach used by workflow management systems, which define are central control-flow that is not adaptable during run-time and local being the flexible paradigm pursued by groupware systems. The central problem, which is addressed in his research, is that "BPM systems force companies to implement either centralized or local decision making, instead of allowing for an optimal balance between the two" [29].

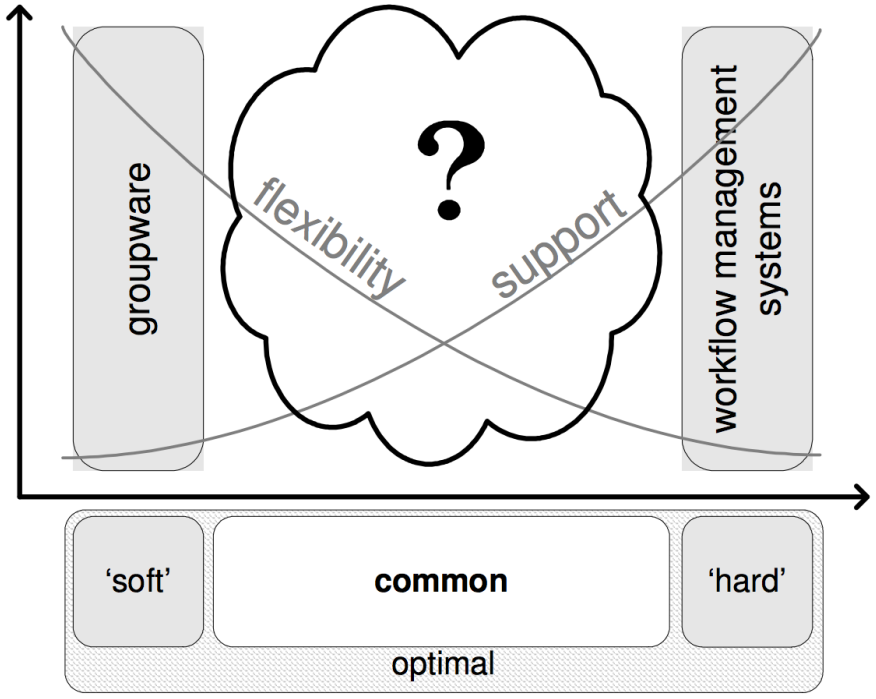


Figure 3.1.: Finding the optimal combination of flexibility and support [29].

Pesic continues his work by looking at the two paradigms from another perspective, perceiving the control-flow based approach as the traditional and a less restrictive, constraint-based approach as the more recent one.

The traditional approach uses a predefined control-flow that is within the limits of what is possible, but also restricts the users by not allowing any variation that makes use of resources outside those boundaries. This is an example for a strictly

imperative paradigm, which can be found in various contemporary workflow management systems.

In contrast to the traditional approach, the constraint-based approach does not define the process using a pre-specified control-flow. It is rather a lose boundary which only specifies a set of rules that define what is possible and what is forbidden. While the forbidden activities can still be considered a boundary, the constraint-based approach offers a lot of flexibility, as the user is able to make local decision within the realm defined by the possible constraints, which are included, and the forbidden activities, which are excluded from execution.

This analysis supports the view that current solutions for business process modeling do not meet the needs of today's knowledge-intensive processes. They either support a very strict, control-flow based approach or a flexible one, which does not provide any support to the user. One of the main objectives addressed in the following chapters is to find an optimal ratio between those two opposing forces.

3.3. Complexity Measurement of CMMN

When analyzing business process modeling languages, complexity is an important figure and can be considered an indicator of the applicability of the language in an actual business environment.

As already mentioned, Fahland et al. discussed the issue of understandability of process modeling languages. They criticize, that there has not been much research on issues such as model understanding and model complexity [9]. Yet, some authors claim the superiority of one modeling language over another. An example for this is [28], which claims that BPMN has advantages over UML Activity Diagrams simply because it "is more conducive to the way business analysts model". Fahland et al. mention that this assumption is not necessarily incorrect, but argue that it is not based on a specific theory to support these kinds of statements (cf. [9]).

One possible solution to this issue of properly analyzing a business process modeling language is to analyze its complexity and compare it to the complexity of other languages competing in the same field. According to Siau and Rossi [38] there are empirical and non-empirical techniques suitable for the analysis of modeling languages. A non-empirical technique was proposed by Rossi and Brinkkemper in [36]. Their meta-model-based method complexity analyses a process modeling language by calculating the complexity of a model evaluating its meta-model.

Marin et al. [22] utilized this approach in order to compare the complexity of CMMN with similar languages such as BPMN 1.2 and UML Activity Diagrams. They argue that the majority of process modeling methods have overlapping functionality, which results in a set of options modelers of business processes have. Even though CMMN addresses case management in particular, the authors conclude that it is still important to measure the complexity of CMMN and compare it to other modeling techniques.

In order to calculate the complexity of the CMMN 1.0 specification, the authors use the same subset of Rossi and Brinkkemper [36], which was already used by Indulska et al. [17] to calculate the complexity of various subsets of the BPMN specification and Recker et al. [34] to compare the complexity of BPMN with the one of UML. Formula 3.1 was used to calculate the cumulative complexity.

$$C^{'}(M) = \sqrt{n(O_M)^2 + n(R_M)^2 + n(P_M)^2} \tag{3.1}$$

This formula calculates the complexity using the number of objects, relationships and properties contained in the meta-model of the process modeling language with:

- $n(O_M)$ being the number of objects in the method M
- $n(R_M)$ being the number of relationships in the method M
- $n(P_M)$ being the number of properties in the method M

The authors extracted 39 object types, four relationship types and 28 property types. Using Formula 3.1 this results in a cumulative method complexity of 48.18.

This complexity was used by Marin et al. to compare CMMN 1.0 to other business process modeling languages that have been evaluated using the same method. As various subsets of BPMN 1.2 have been created for specific use-cases, they have been included in the comparison. These are the BPMN versions by the U.S. Department of Defense [41], as well as the subsets analyzed by zur Muehlen and Ho in their case study [49] and the frequently used objects extracted by zur Muehlen and Recker [50]. The results are shown in Table 3.1.

They show that BPMN 1.2 is the most complex of the modeling languages evaluated, while EPCs and UML Activity diagrams are the least complex. The various subsets of BPMN show that they manage to reduce the level of complexity. Most importantly, CMMN 1.0 has a lower complexity than any of the four versions of

Method	**Objects**	**Relationships**	**Properties**	**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
EPC	15	5	11	19.26
UML 1.4 Activity Diagrams	8	5	6	11.18

Table 3.1.: Cumulative complexity of different business process modeling methods. Author's own compilation based on Marin et al., 2014 [22]

BPMN which can be considered an indicator for an improved understandability of CMMN.

The results are visualized in Figure 3.2, which shows all process modeling languages, including the various subsets of BPMN 1.2, along the three dimensions. This illustration highlights the difference in complexity between the different approaches and shows the general reduction achieved by creating subsets of a notation. Nevertheless, these subsets of BPMN 1.2 still find themselves in the center of the cube whereas the languages with the least complexity are found in the lower left area of it.

While these results are a good way to get an understanding of the complexity reduction, they can only be considered a starting point for further analyses. One of the reasons for this is the difference of the languages compared. While sharing some of the functionality there are key aspects of each language that distinguish it from the others. Such differences might not get recognized using the metrics applied in this approach.

Furthermore, the authors state that future research could focus on the extraction of subsets based on the CMMN 1.0 specification (cf. [22]). This is one of the key aspects of this thesis and will be discussed more thoroughly in the following chapters.

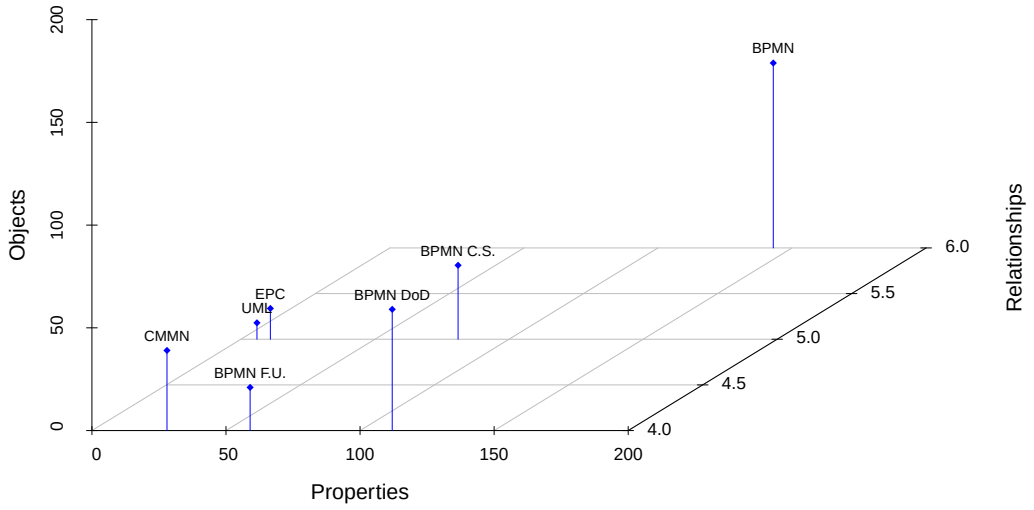


Figure 3.2.: The cumulative complexity of CMMN compared to the other process modeling notations in Table 3.1 in an Object-Relationship-Property cube. Author's own compilation based on Marin et al., 2014 [22].

Part II.

Supporting Knowledge-intensive Processes with CMMN

4. Workflow Patterns

In their work on what the authors refer to as *Workflow Patterns*, van der Aalst et al. [43] emphasize the importance of a common way to compare different business process modeling techniques. They argue that "if workflow specifications are to be extended to meet newer processing requirements, control flow constructors require a fundamental insight and analysis" [43]. Using this analysis, this chapter focuses on the summary of basic workflow patterns as described by the authors and the extraction of patterns that need to be part of an application supporting the knowledge-work lifecycle. While the area of case management is focused more on flexibility compared to contemporary workflow management systems, there are still basic patterns, which need to be part of a software solution that supports knowledge workers.

The following sections analyze some of the different patterns described by the authors. Whenever possible, the same terminologies and methods of categorization were used. These patterns can have different types of complexity and "range from fairly simple constructs present in any workflow language to complex routing primitives not supported by today's generation of workflow management systems" [43].

The main objective of this chapter is to use the analysis of different workflow patterns in order to find a suitable subset of constraints to meet the requirements of an application supporting Knowledge-intensive Processes. The two forces influencing these requirements are the reduction of complexity on the one hand and the creation of flexibility on the other. While these are not opposing forces, there are some cases in which reasonable compromises are inevitable.

4.1. Control-Flow Patterns

In [37] Russel et al. mention the disparity between the control-flow patterns specified by a modeling language and the number of available patterns within a software

implementation. As this section focuses especially on the original control-flow pattern defined by van der Aalst et al. and the Workflow Patterns Initiative [43] it is important to consider the ongoing changes concerning the different patterns. In [37] this issue is addressed and the authors try to incorporate all the changes and additions made to the original control-flow patterns.

The following sections analyze the available control-flow patterns in detail and use the same means of categorizations as the authors of the original papers.

4.1.1. Basic Control-Flow Patterns

Basic control-flow patterns are typically very simple constructs, which users usually expect to be available in any workflow management system. "They define the basic modeling patterns of business processes" [47]. The most basic example for such a construct is the creation of a sequence between multiple tasks, which results in the creation of a hierarchy. A task, which has a predecessor, can only be executed once the previous one has been completed. Basic constructs like this are discussed in this secion.

Sequence

A Sequence, which is referred to as *Sequential Routing* in the specification of terminologies by the Workflow Management Coalition [48], specifies the ordering of activities in a sequential or hierarchical order. This pattern can be regarded as the most basic one, representing a natural ordering of activities. In terms of CMMN this would be the sequential ordering of *Tasks* or *Stages*, which can both be regarded as activities, as they both contain information on what specific activity is to be performed.

A motivation for the usage of this pattern is explained by Russel et al.: "The Sequence pattern serves as the fundamental building block for workflow processes. It is used to construct a series of consecutive activities, which execute in turn one after the other" [37].

Figure 4.1 shows the implementation of a sequential structure using CMMN. The notation uses Connectors and Sentries in order to create this basic control-flow.

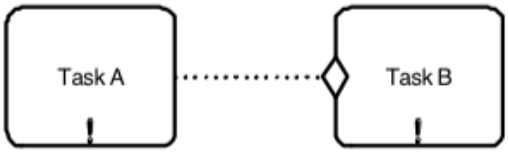


Figure 4.1.: Sequential ordering of two tasks using CMMN as specified by the documentation of the Object Management Group [26].

Parallel Split

A Parallel Split pattern enables the concurrent execution of more than one activity, by splitting the order of execution after the completion of an activity. It is also referred to by [48] as *AND-Split* due to the possibility to execute two activities in parallel. It is defined "as [...] a mechanism that will allow activities to be performed concurrently, rather than serially. A single path through the process is split into two or more paths so that two or more activities will start at the same time" [47]. According to [11] and [37] implicit as well as explicit implementations of this pattern can be found. Both methods are used in recognized process modeling languages.

Synchronization

In order to synchronize two concurrent sequences, the Synchronization pattern or *AND-Join* [48] combines multiple control-flows into one subsequent activity. Just like the Parallel Split divides the process into two or more branches, the Synchronization pattern is used to combine them at a later stage.

According to Atwood's analysis of Workflow Patterns, "the Parallel Split and Synchronization pattern speeds up the process by having the instance travel all the parallel paths through it simultaneously" [1].

Multiple Choice

While the Parallel Split pattern focuses on the concurrent execution of multiple branches, the Multiple Chose pattern does not need all branches to evaluate as completed. It is also referred to by [48] as *OR-Split*, since either branch may allow the overall process to continue. Nevertheless, this pattern is not restrictive as it also allows multiple branches to be executed (see Exclusive Choice Pattern). According to[43] the Multiple Choice pattern belongs to the group of patterns that do not

have full support in all workflow engines, but occur frequently in real-life business scenarios.

Multiple Merge

In order to provide a way of joining mutliple branches together, which were previously split by the Multiple Choice pattern, the Multiple Merge is used. "Sometimes two or more parallel branches share the same ending. Instead of replicating this (potentially complicated) process for every branch, a multi-merge can be used" [43].

Exclusive Choice

While the Parallel Split pattern does not specify which branch to execute and thus requires the execution of all branches if necessary, the Exclusive Choice or *XOR-Split* [48] creates multiple subsequent branches. It is only allowed for one of the branches to be marked as completed for the process to continue with the successive activity. The other parallel branches are no longer considered.

While the Parallel Split or the Multiple Choice patterns are not restrictive by allowing various branches to be executed, the XOR-Split is an exclusive pattern, limiting the execution of the process to just one branch. "The pattern is exclusive in that only one of the alternative paths may be chosen for the Process to continue" [47].

Simple Merge

The Simple Merge resembles the join operation for a previously instantiated XOR-Split, which is why the Workflow Management Coalition refers to this pattern as *XOR-Join* [48]. It enables two or more branches to collectively merge into a subsequent activity. Just like the Exclusive Choice, this pattern is restrictive if compared to the Multiple Merge due to the exclusive choice of one branch.

4.1.2. Advanced Control Flow Patterns

In addition to the basic control flow patterns described in the previous section, there is also a large number of advanced control flow patterns that have been extracted by van der Aalst et al. in their work on Workflow Patterns. They state that "as opposed

to the [basic control flow patterns], these patterns do not have straightforward support in most workflow engines" [43].

Nevertheless, some of them are especially interesting for environments that are data-centric and unstructured. The patterns belong to semantic groups which are explained below:

Structural Patterns

The structural patterns impose different restrictions on workflow models. According to [43] the decision whether or not to allow the creation of complex structures such as cycles and implicit termination patterns. can be a challenging task. The authors argue that "a real issue here is that of suitability. In many case the resulting workflow may be unnecessarily complex which impacts end-users who may wish to monitor the progress of their workflows" [43].

With regard to the issue of complexity, which will be addressed towards the end of this conceptual part of the thesis, such structural patterns can be regarded as unsuitable for Knowledge-intensive Processes where the focus is on flexibility and not restrictions.

Multiple Instances

The ability to allow multiple instances of activities is another area of patterns anaylzed by the authors. It can be regarded as a concept, which "corresponds to multiple threads of execution referring to a shared definition" [43]. The possibility to allow multiple instances is a vital part to allow the proper support of the knowledge-work lifecycle, which involves the extraction of patterns from multiple instances of a process with the same definition (template).

The software prototype used for the implementation natively supports multiple instances by providing features such as template and sub-page creation. This will be analyzed further in the implementational part of this thesis.

State-based Patterns

State-based patterns should be regarded as important, since their focus is on the analysis of the state of a process. With a focus on data-centricity the notion of an activity's state, such as *contains-data* and *not-contains-data*, is an important mechanism also commonly used in computer science (cf. [43]).

The most important pattern belonging to the group of State-based Patterns is the Milestone, which is used to mark distinct sections of a process and to calculate the current progress. The milestone is also a core part of the CMMN specification and its visual notation. In the following chapter a different approach will be proposed to allow the tracking of the progress imitating the milestone pattern.

Cancellation Patterns

Cancellation patterns allow the termination of activities within the workflow. In [43] the authors distinguish between the cancellation of an activity, which can be compared to a task or stage in CMMN, and the cancellation of a whole case. While the former can be regarded as overly complex if connected to constraints, the simple removal of a previously added activity should be regarded as a crucial functionality. Cancelling an entire case on the other hand gives users the freedom to control the progression of a case and its termination, making it important as well.

4.1.3. Important Control Flow Patterns

The control-flow patterns below represent the basic requirements proposed by the author of this work. The list contains the different patterns explained in the previous section, that should be regarded as a minimal subset of all available patterns that workflow management systems, with a strong focus on runtime flexibility, should contain. Reasons for the choice of the different patterns are also provided in order to explain the decisions made.

CFP1 - Sequence

A sequence is one of the most basic constructs in a process. It can be used to put different activities into order and should be be part of any process modeling software that allows the structuring of information elements. "The Sequence pattern is widely supported and all of the workflow systems and business process modelling languages examined directly implement it" [37].

Reason: It is important for structuring multiple activities and offers many ways for users to structure different units of work. It can also be found major process modeling software and easy for end-users to understand.

CFP2 - Parallel Split

Splitting a sequence into two or more flows that are executed simultaneously is another pattern, which is often used and can be found in the major process modeling specifications. Russel et al. mention that this pattern is used in both an explicit and implicit way (cf. [37]).

Reason: The parallel split allows the concurrent execution of tasks and can be found in all major process modeling tools. It is also a simple construct easy to understand by end-users.

CFP3 - Synchronization

The Synchronization pattern is closely related to the Parallel Split and explicit as well as implicit implementations can be found (cf. [37]). It allows the creation of complex logical structures.

Reason: It is closely related to the Parallel Split and a logical partner for creating basic control-flows.

CFP4: Multi-Choice

The Multi-Choice pattern is also found in most process models and can be integrated in an explicit or implicit way. It involves more logic than the Parallel Split pattern, as the execution of one branch is sufficient to complete a whole set of activities.

Reason: Closely related to a Parallel Split and usually part of other process modeling techniques.

CFP5: Multi-Merge

Merges two or more branches of execution into one common successor. Just like the Synchronization pattern it can be found in all major process models and is implemented in explicit and implicit ways.

Reason: Used in combination with the Multiple Choice pattern and, like the Synchronization pattern, allows for the creation of a more complex logic.

CFP6: Implicit Termination

The Implicit Termination pattern allows for any branch within the process to reach a final state without the need for all other parallel branches to be terminated. It can be regarded as counterproductive for activities in a knowledge-intensive environment if different paths of work cannot be completed.

Reason: With regard to flexibility the Implicit Termination pattern is a natural pattern and represents a behaviour that a Knowledge Worker is likely to expect.

CFP7: Multiple Instances without Synchronization

The Multiple Instances without Synchronization pattern allows the same activity to be part of multiple instances of the process. This pattern can be compared to the creation of a template which is used for multiple processes that share common features. While patterns exist that include synchronization, this pattern explicitly disallows it. Consequently, the state of each instance is not tracked by the siblings and does not influence their state.

Reason: With regard to the creation of templates and the usage of process patterns which include best-practices extracted from previous process iterations, the ability to run multiple instances of a process can be regarded as crucial for knowledge-intensive environments.

CFP8: Deferred Choice

The Deferred Choice pattern represents a stage throughout the process in which the user explicitly decides to execute or leave out a specific branch. By doing so, the user influences the control-flow ad-hoc.

Reason: In order to allow the flexible execution of a process with only very little restrictions imposed on the user, it is important to promote individual actions.

CFP9: Milestone

The Milestone pattern requires the representation of different states to be used adequately. However, this can be achieved by using the indicator of completion for different activities to determine the state of a stage within the process.

Reason: While the concept of a Milestone can be implemented explicitly, there are also ways to support this pattern indirectly, which does not require the representation of states.

CFP10: Cancel Case

The Cancel Case pattern allows for a case to be removed entirely, with all running instances being terminated. While this is an important feature with regard to creating a lean environment which only contains useful data, this feature is designed for advanced users. Once a modeling expert recognizes that a process is no longer needed, he can decide to erase it from the environment.

Reason: Important feature required by advanced users to control the cases available to regular users.

4.2. Unsupported Patterns

With regard to the reduction of complexity and the specific requirements imposed by Knowledge-intensive Processes, there are many patterns, which are not supported. In many cases these unsupported patterns require the creation of complex processes, which should be regarded as disadvantageous, due to the various types of users working with a workflow management system handling unstructured and adaptive cases.

Furthermore, the selection of important patterns introduced in the previous section should not be regarded as exhaustive, but rather an initial proposal of basic functionality, which can be useful for case management. Some of the patterns presented in this section need to be reconsidered once the developed prototype

has been evaluated further. The following is a compilation of the most significant patterns, which have not been selected as part of the process modeler developed in this thesis.

Cycles

Cycles are a powerful control-flow pattern, which allow the creation of complex processes with repetitive activities. The logic is commonly defined by complex expressions, which specify the truncation condition. Due to their complexity, cycles can easily lead to deadlock scenarios in which a process gets stuck in endless loops, making the continuation impossible. Thus, the ability to create cycles can be regarded as a feature only available to expert users with advanced knowledge in process modeling. However, the possibility for regular Knowledge-Workers to create their own worklists requires a common approach, shared by end-users and experts. For that reason, the creation of cycles is explicitly disallowed in the implemented prototype.

Complex Instancing

Other modeling languages often make use of complex instancing feature which allow modelers to specify not only the process itself but its various instances. In most cases these patterns require a design time knowledge, in order to specify the number of instances which can exist. With a strong focus on the ability to constantly adapt a process, the ability to pre-define instances is counterproductive and does not create any major benefits for Knowledge-Workers. Other instancing patterns do not need a definition at design-time (cf. Pattern 15 in [43]), but use complex logic to define the number of instances at run-time. CFP7 represents an exception due to the flexibility it offers. It allows multiple instances of a case to exist and does not require any more logic, as it does not attempt to synchronize processes executed simultaneously. Furthermore, many of the instancing patterns also allow the existence of multiple instances of single activities. In the implemented prototype, the focus lies on the simultaneous execution of several instances of a whole process, while each activity within a process can only have on instance.

5. Requirements for Knowledge-intensive Processes

Requirements Engineering plays a vital role within the development process of a new software solution. Broy [2] supports this view and emphasizes its importances, especially when developing novel and software-intensive systems. As the developed prototype represents a new software system without a predecessor, the thorough analysis of the requirements to support Knowledge-intensive Processes is important. This chapter focuses on this analysis with a strong focus on the functional requirements of a process-modeling application supporting case management.

In addition to the support of Knowledge-intensive Processes, it is important to analyze the specific requirements for the lifecycle of Knowledge Work. Based on the research performed throughout the course of this thesis, Knowledge-intensive Processes and the cases, which handle them, undergo a constant adaption in which they are redefined, modified and improved. The basic idea behind this is the perception that throughout this constant alteration process, patterns can be extracted that help modelers to create process templates which can be used for similiar cases.

5.1. Analysis of Requirements

As mentioned before, there has been a lot of research focusing on knowledge-intensive processes. In many cases, researchers presented their own compilation of requirements for fostering these types of processes in a run-time application supporting knowledge workers. In order to come up with a reasonable list of requirements for the application developed in this work, this section analyzes the key components found in the literature.

In their work on knowledge-intensive processes, Ciccio et al. [6] present a compilation of important characteristics and requirements of these processes. This

compilation is then used to evaluate the power of currently available workflow management systems concerning the handling of knowledge-intensive processes by comparing the features of each solution to the requirements. The authors come to the conclusion that "the characteristics and requirements of KiPs force to reconsider the classical process life cycle based on the design execute & monitor analyze re-design sequential steps. The boundary between process design and execution gradually disappears, replaced by a continuous interleaving and overlapping between design, execution and adaptation activities" [6]. They also mention that none of the software solutions evaluated fulfilled all the requirements compiled.

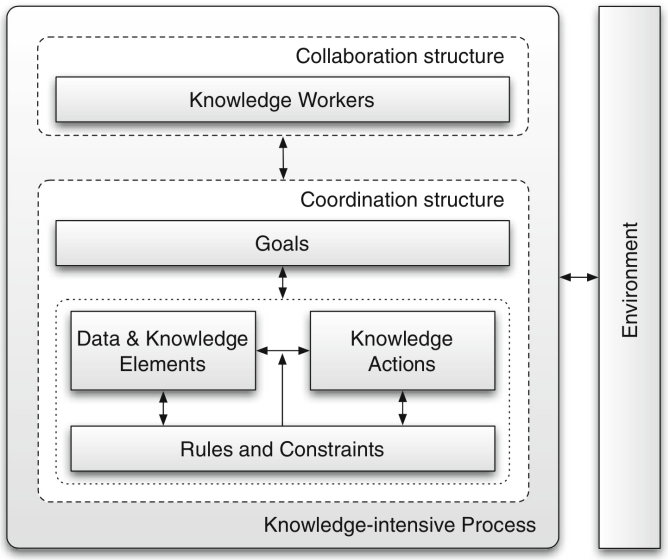


Figure 5.1.: The fundamental components of Knowledge-intensive Processes as analyzed and illustrated by Ciccio et al. [6].

Figure 5.1 shows the fundamental components of Knowledge-intensive Processes as proposed by the authors. It shows the "tight integration of *data & knowledge elements* with *knowledge actions*" [6]. This information model is influenced by *rules & constraints* which are often based on guidelines or best-practices. The *goals* specified by knowledge-workers relate to the elements specified in the informational model and are influenced by the completion of the actions specified and the evolution of data and knowledge. Furthermore, the *environment* constantly changes all elements of a Knowledge-intensive Process.

The following is an analysis of the key requirements for supporting Knowledge-intensive Processes in a software system based on these findings. Support for the

different requirements found in the literature is provided along with a detailed description of the specific attributes.

5.1.1. Flexibility

An important requirement which is often mentioned in the related literature is the flexibility which is required when working in knowlegde-intensive environments. This need has been well-covered in the previous chapters and can be regarded as one of the main drivers for new approaches in the field of Adaptive Case Management.

In their own proposal for a framework supporting case management in social networking environments, Nezhad et al. [23] reiterate the importance of flexibility within their solution. They aim to achieve this by making important elements such as tasks and templates adaptive and by allowing the adding, skipping and removing of tasks.

This need for a constant ability to control and influence a business process that is knowledge-intensive is also supported by Herrmann and Kurz [14] in their work on Adaptive Case Management. They mention that the flexibility-to-use, which describes ”whether the system is able to cover new business requirements without a major change” [14], is not adequately supported by current BPM systems.

In summary, the flexibility in a knowledge-intensive enviroment is very important. While this can be partially achieved by adding adaptive features to certain aspects of an application, the focus should be on the overall provision of flexibility throughout all stages of a process, which includes its design-time and more importantly also its run-time phase.

5.1.2. Data-centricity

One of the main differences between the handling of Knowledge-intensive Processes and contemporary workflow management systems is the focus on *data objects* rather than the state of the workflow. This perception is supported by van der Aalst et al. who state that ”in contrast to existing workflow management systems, the logistical state of the case is not determined by the control-flow status but by th presence of data objects” [45].

The objective behind this approach is the idea that the primary driver in a knowledge-intensive environment is the presence and absence of different data objects. Instead of determining a process’ state by looking at the activities, the value

of these data objects suffices to analyze the progress. Thus, data-centrictity is vital to the proper handling of a case and should be a key concept in an application supporting Knowledge-intensive Processes.

5.1.3. Goal Definition

As already discussed in Section 2.2 of Chapter 2 on Declarative Processes, a case in a knowledge-intensive environement should be driven by the goals which lead to its completion and not the specific activities that are required to get there. Furthermore, Tran et al. [40] argue that a Knowledge Workers performance is driven by the goals that have been defined. In other words, the process is required to be declarative.

This view is also supported by Fahland et al. in their second proposition for the nature of a process modeling language: "Given two process models, establishing circumstancial information will be easier on the basis of the model that is created with the process modeling language that is relatively more declarative in nature. Establishing circumstancial information will be easier on the basis of a declarative process model than with an imperative process model" [9]. As the information handled in Knowledge-intensive Processes can be regarded as mainly cirucmstancial, the use of a declarative model is vital. With regard to the definition of goals, the declarative model is more useful as it builds on this principle.

Even though a case is usually defined by an overall goal, there can be subgoals which focus on a specific part of the process. Such goals are tied to a specified set of elements which need data-input in order for it to be reached.

The ability to specify goals can be achieved in various ways. Specifically, it can be done in an explicit or implicit way, either making use of constructs such as milestones or simply using structural elements to create a goal-like environment. For the implementation of a software system, either method can be regarded a possible solution.

5.1.4. Reduction of Complexity

Another important aspect to consider when looking to implement a software system which supports Knowledge Workers is the fact that the majority of users will not have any modeling expertise. It is therefore vital to the successful integration of such an application in a working environment, to consider method complexity and the capabilities of its end-users.

In [9] Fahland et al. mention the importance of understandability in process modeling environments. Especially in environments with a majority of circumstancial information and ad-hoc activities performed by users, there should be a focus on a declarative model instead of making the process logic overly complex internally. The authors reconfirm this by stating that "after all, not only designers are reading process models but end users too" [9].

With the main objective of this thesis being the empowerment of end-users to support Knowledge-intensive Processes, the reduction of complexity plays and important role when analyzing the functional requirements for a software solution. Due to this importance, Chapter 7 presents the actual complexity measurement of the developed prototype, similar to the work performed by Marin et al. in [22].

5.1.5. Support of Constraints

With a strong focus on reducing the complexity of the developed prototype with regard to modeling expertise, the support of constraints which can be applied to the process needs to be conservative. Nevertheless, it is vital to the power of the modeling tool to be capable of handling basic constraints which enable modeling experts to make use of different techniques to add logic to the process when necessary.

Even though on the of the main objectives for creating an environment for the handling of Knowledge-intensive Processes is to consider end-users and their requirements at all times, the view of a modeling expert needs to be regarded as well. This expert needs to be enabled to extract important information created by regular users in order to generate process logic whenever a pattern is encountered. A CMMN-specific analysis of the possible constraints is discussed in a more detailed fashion in the following chapter.

5.1.6. Roles

With regard to the problem of combining different views such as the one of a typical knowledge worker and that of a modeling expert, a consistent and powerful role management system is crucial. Van der Aalst et al. specifically mention the importance of a role handling mechanism in [44].

The presence of these two distinct roles does not reflect the complete environment of a Knowledge-intensive Process. Usually, there are many different types of roles

which need to be considered, with many of them requiring different views on the process.

While the adding of roles can be regarded as a main feature, flexibility remains a core requirement. In order to not restrict the users of a software system in any way, the usage of a complete role model should be optional and at the end-users discretion.

5.2. Functional Requirements

Using the analysis of requirements above, this section gives reasons to justify in how far each requirement can have a positive impact on the implementation and its way of supporting knowledge workers. While more requirements will be extracted throughout the following chapters, the four basic ones presented here should be regarded as important, since they reflect the core principles of Knowledge-intensive Processes and Adaptive Case Mangement in general.

The following is a list of specific requirements compiled by the author. While CMMN-specific requirements are analyzed in the following section, this list represents the most important functional requirements the developed prototype should be based on.

Flexibility at run-time

Due to the constant adaption of a case and the specific need to be able to alter information at any given time, the process needs to be handled in a flexible way. In order to accomplish this a case needs to evolve over time, allowing its users to add, remove, and edit data throughout all stages (e.g. design-time, run-time).

Data-elements defining a case

In a knowledge-intensive environment, the state of a case is defined by data being absent or available. The process should not only support the adding of data elements, but should be built around the presence of different types of data elements.

Declarative case definition

The definition of a case should be done in a declarative way using goals instead of specific pre-defined activities to determine its objective. With regard to an implementation, this requires the system to be designed in a non-restrictive way allowing Knowledge-Workers to address a problem (case) in an indepedent way. In other words, guidance should not be achieved by restricting the users actions to a control-flow but solely by the goals specified.

Role management system

The role management system needs to consider the presence of various users with different intentions. It also has to provide mechanisms to restrict the access of regular Knowledge-Workers to the tools designed specifically for expert users with modeling capabilities.

The requirement to support constraints as well as the non-functional requirement to reduce complexity is not covered in this list. Both of them are discussed in detail in the following two chapters, which generate a suitable subset of CMMN including constraints (Chapter 6) and calculate the complexity of the developed application (Chapter 7).

6. Extraction of a Suitable Subset

The aforementioned extraction of a subset of the CMMN specification is an important task and resembles one of the main goals of this work. Marin et al. mention this in their paper which analyses the complexity of CMMN: "Another venue for future research is to identify subsets of the CMMN notation. As process modelers begin to use CMMN, it will be useful to identify the subsets of the specification that start to emerge" [22]. They also state that this can be done in the same fashion as the creation of subsets of BPMN 1.2 by zur Muehlen et al. in [49], [50], [51].

Using the requirements analyzed in the previous chapters, it is now possible to propose a suitable subset of the CMMN 1.0 specification in order to support Knowledge-intensive Processes. The subset can be regarded as the basis for the implementation undertaken in this thesis.

6.1. Structuring Activities with Tasks and Stages

One of the major concepts for structuring activities in CMMN is the use of tasks, as "an atomic unit of work" [26], and stages to help the users with the planning and grouping of the work units (cf. [26]). In order to be able to implement patterns such as sequences (CFP1) and hierarchies, the concept used in CMMN accomplishes this with just two different types of elements. While in theory the same patterns could be created only using tasks, the availability of a structural element (Stage) helps grouping and organizing different work items.

6.1.1. Human Tasks

As already mentioned in the introduction of CMMN, HumanTasks are a way to specify tasks that are specifically executed by a case worker. They stand in contrast to CaseTasks which are discussed in the following section. As case workers will usually start structuring a knowledge-intensive process by adding units of work

which they extract throughout the execution of this process, the HumanTask is an important way for users to add the most atomic units of work to the worklist.

HumanTasks are an essentail feature of CMMN and should be part of any subset as they represent atomic units of work and explicitly refer to the action of a knowledge worker.

6.1.2. Case Tasks

According to the CMMN specification, CaseTasks are used "to call another case" [26], which is an important feature to structure even the most complex processes. As soon as a case is maturing, it is very likely that certain aspects of it are so complex in nature, that it makes sense to create an entire case just for them. Through the use of a CaseTask, a case can be refered to directly within another case.

6.2. Creating Relationships

Even though there are various scenarios in which a knowledge-intensive process does not need any relationships due to a majority of tasks which are executed by knowledge workers independently, it is important that end-users as well as modeler have ways of bringing different tasks into relation. This section analyzes how the features to create relationship between entities provided by CMMN can be added to the subset created in this chapter.

In CMMN any relationship is represented by a *Connector* and a *Sentry*. While the connector simply indicates the existence of a dependency, the sentries indicate which element requires input from another one.

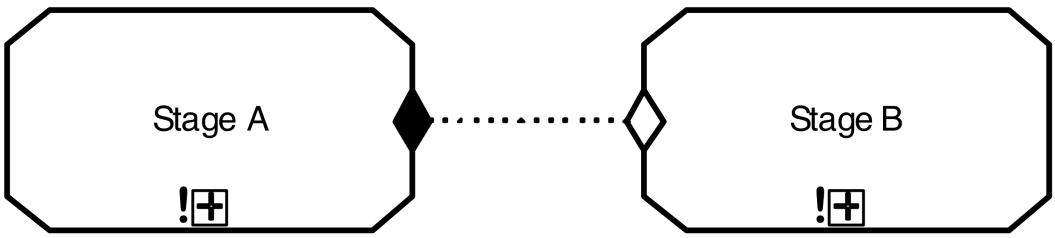


Figure 6.1.: Dependency between two Stages in CMMN using Connectors and Sentries. [26].

6.3. Adding Constraints

The previous sections already analyzed the elements necessary to structure the process data in different ways and provide methods for their grouping and ordering. However, there are certain constraints which modelers should be able to use in order to create a process that contains logic which helps to guide end-users to complete their tasks in an adequate manner on the one hand, but also provides enough flexibility to leave most of the decision making to the knowledge-workers.

The following is a compilation of the constraints extracted by the author, which are based on the previous analysis of Workflow Patterns on the one hand as well as the basic requirements listed in the previous chapter.

C1 - Completion of a simple task

There are certain tasks that represent an atomic unit of work, which does not require the input of any data. Examples for such tasks could be simple activities such as *Inform supervisor* or *Review document*, which do not contain any attributes. The completion of these tasks is defined by a single attribute which represents its binary state (incomplete / complete).

C2 - Completion of a task with attributes

In contrast to the simple task in C1, tasks can also have one-to-many attributes attached to them. These attributes can be filled with values of various types and are usually maintend by users currently working on the related task. The completion of the task is defined by the state of the underlying attributes. As soon as all attributes have been filled with information, the task should be considered compelte. In order to prevent various forms of dead-locks, certain users need to be able to skip and redo tasks.

C3 - Hierarchical organization of cases and tasks

The *CaseTask* which has already been explained at the beginning of this chapter, refers to a separate case. Using this feature, complex structures can be created by referencing entire sub-processes. The completion of a *CaseTask* is triggered by the completion of its underlying case.

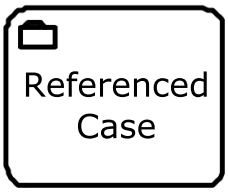


Figure 6.2.: The representation of a CaseTask in CMMN used to reference another case based on [26].

C4 - Structuring tasks with stages

The CMMN spefication refers to stages as structural objects which can be used to group and organize tasks. While it would be possible to create a different case for each sub-goal represented by a stage, it is often necessary to define different phases of a case. By allowing a stage to contain one-to-many Tasks, this dependency can be created.

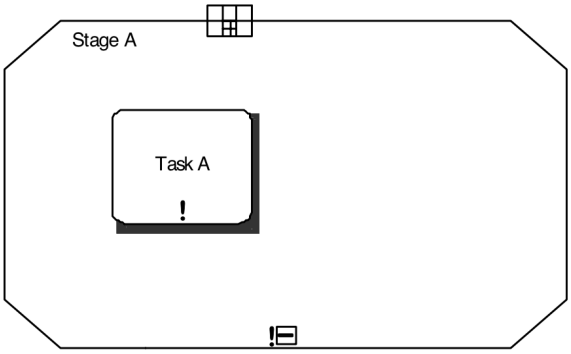


Figure 6.3.: A Task added to a Stage in CMMN used to order activities and create hierarchies [26].

C5 - Producer-Consumer dependency between tasks and stages

In order to be able to create a sequential ordering of tasks and stages, which has been analyzed as a core Workflow Pattern, the Producer-Consumer dependency is used. Figure 6.4 shows how this dependency is created between two tasks in CMMN. While this serves as a visual representation of the relationship between elements, there needs to be additional logic concerning the order of execution. If an element is the consumer of another element, it requires

input from its predecessor. Thus, the consumer can only be executed once the producer has been completed. Skipping a task will have the same effect as completing it. With regard to CFP1 this constraint is required to allow the proper usage of the sequence pattern.

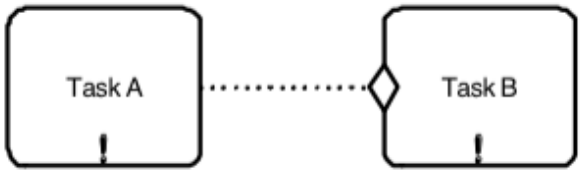


Figure 6.4.: A Producer-Consumer dependency between two tasks [26].

C6 - Multiple consumer tasks

In addition to a simple Producer-Consumer dependency, there exist various scenarios in which multiple consumers are dependent on a single consumer. All of the consumers are linked to the producer using a single connector shown in Figure 6.4. Using the same logic explained in C5, each consumer task should only be triggered once the producer task has reached the state of completion. This constraint represents a combination of CFP1 and CFP2, putting multiple tasks into a sequence and allowing the splitting into different branches. Parallel execution is not handled by this constraint, as it is part of the AND-Join described in C7.

C7 - AND-Joins between tasks

The AND-Join, which is also part of the aforementioned control-flow patterns, is an important constraint which enables modelers to add a more complex logic to processes. Dependencies can be created in which a task requires more than one action to be completed before it can be executed.

While the AND-Join should be regarded as an essential tool for process modelers, it comes at the price of user flexibilty. This is an important aspect to consider before the creation of complex dependencies.

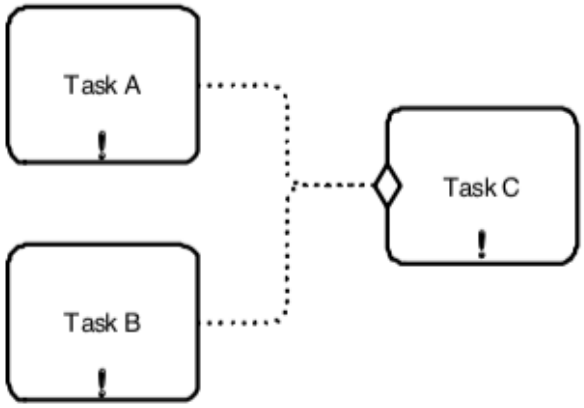


Figure 6.5.: An AND-Constraint created between two Producer tasks and one Consumer task [26].

C8 - OR-Joins between tasks

Similar to the AND-Join, the OR-Join is also part of the control-flow patterns, and can be regarded as equally important. An OR-join specifies that only one of the preceeding tasks needs to be completed in order for the control-flow to continue. It is not as restrictive to users as the AND-Join, since all possible combinations of execution are allowed.

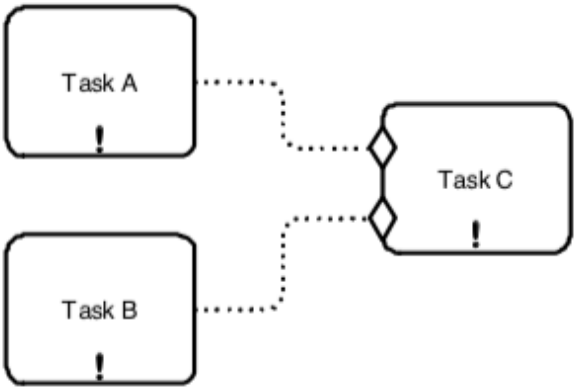


Figure 6.6.: An OR-Constraint created between two Producer tasks and one Consumer task [26].

C9 - Dependency between tasks and stages

While dependencies between two tasks or two stages represent trivial dependencies, relationships between a task and a stage need to be possible as well. A single task (e.g. a CaseTask) could be the predecessor of a whole stage and thus needs to be executed beforehand. While the handling of this type of constraint is identical to a normal connection between two elements of the same type, it creates a more complex logic regarding the order of execution.

7. Complexity Measurement

Complexity measurement plays an important role when analyzing different process modeling languages. In Section 3.3 of Chapter 3 the complexity measurement of CMMN according to Marin et al. [22] showed that the complexity of the CMMN 1.0 specification is much lower than the complexity of BPMN 1.2 and its various subsets. In fact, the CMMN 1.0 specification (48.18) had a cumulative complexity more than three times lower than the one of BPMN 1.2 (169.07). This put CMMN 1.0 between BPMN 1.2 and the less complex Event-driven Process Chain (EPC) and UML 1.4 Activity Diagrams. With the perception that CMMN 1.0 is still quite complex, the focus during the definition of a suitable subset was to create a lighter version of the full specification. In order to analyze in how far this goal was achieved, it makes sense to calculate the cumulative complexity for the prototype developed in this thesis using the same method. This chapter focuses on the calculation of this complexity and documents the different steps performed throughout this approach.

7.1. Analyzing the Number of Objects

When counting the number of objects contained in the meta-model of the implemented solution, it is important to distinguish between objects that are actually part of the process model and such objects that were generated by a variety of tools (e.g. a persistency library). In the calculation only the former ones were considered in accordance with the approach used by Indulska et al. in [17]. The goal behind this approach by Rossi and Brinkkemper [36] is to calculate a complexity given a number of objects which a potential user of the process modeling application needs to be familiar with. The authors reason that ”a technique with many concepts is more complex to learn, than one with fewer concepts” [36].

The objects in Table 7.1 have been extracted from the meta-model. It contains trivial objects such as a task or a stage, but also different types of Attribute values which represent different object entities. Types, attributes and tasks each have a

	Objects
1	Case
2	Type
3	TypeDefinition
4	Page
5	State
6	AttributeDefinition
7	PageAttribute
8	TaskAttribute
9	TaskDefinition
10	HumanTask
11	CaseTask
12	Task
13	Role
14	Validator
15	Process
16	BooleanValue
17	StringValue
18	IntegerValue
19	DateValue
20	PageValue
21	EnumValue
22	FileValue

Table 7.1.: List of the objects extracted from the meta model.

corresponding Definition object which are also part of the count. The complexity measurement of CMMN 1.0 by Marin et al. [22] mentioned in Chapter 3 counted 36 objects for the specification. With regard to reducing the complexity of CMMN 1.0 using the suitable subset mentioned in the previous chapter, the count of 22 can be considered a first indicator of a reduction in complexity.

7.2. Analyzing the Number of Properties

The counting of properties contained in the meta-model of the developed application does not consider any tool-generated types as well, since they are not encountered at any given time by the end-user. This approach was also used for the counting of objects, but is of paramount importance when indentifying the number of properties as they are generated by tools or only used for internal software specific mechanics more frequently.

Table 7.2 shows the properties which have been extracted from the meta model. Properties such as *name* and *value* occur more than once, because they are part of multiple objects. For example, most visible objects, such as tasks and stages, contain the name property. The number of 26 properties in this model is very close to the number of the CMMN 1.0 meta-model analyzed by Marin et al. [22], which counts 28 properties. Even though the two counts are almost equal, there is still a reduction to be observed, especially because many of the properties are just basic information such as names.

7.3. Analyzing the Number of Relationships

While the extraction of objects and properties can be performed easily using the meta-model, the relationships which are available is not as trivial. The most obvious relation is the linking which is possible between *Tasks* and *Stages* and amongst themselves. Furthermore, it is possible to create constraints between *Tasks*. Two less trivial relationships are the possible connections between a *Page* and a *Type* as well as the definition of *Attributes* that are directly connected to a *Page*. Table 7.3 shows the four relationships which can be found in the implemented prototype. When analyzing the number of relationships, the same methods used when calculating the complexity of CMMN were used in order to allow comparison. The number of possible relationships in all process modeling languages ranges from 4 to 6 (cf. [22]), which indicates the correctness of the method used in this calculation.

7.4. Calculating the Cumulative Complexity

The same formula defined by [36], which was used in Chapter 3, is used to calculate the cumulative complexity:

	Properties
1	name
2	name
3	name
4	name
5	name
6	name
7	name
8	name
9	name
10	value
11	value
12	value
13	value
14	value
15	value
16	value
17	type
18	progress
19	reader
20	editor
21	startDate
22	endDate
23	skip
24	hasPredecessor
25	hasPredecessor
26	validationMessage

Table 7.2.: List of the properties extracted from the meta model.

$$C^{'}(M) = \sqrt{n(O_M)^2 + n(R_M)^2 + n(P_M)^2} \tag{7.1}$$

The authors give the following explanation for their formula: ”The cumulative complexity returns a value that explains the total complexity of the method” [36].

Using Equation 7.1 it is now possibe to calculate the cumulative complexity of

	Properties
1	StageLink
2	PageTypeLink
3	PageAttributeLink
4	TaskConstraint

Table 7.3.: List of the relationships extracted from the meta model.

the process modeling used within the Darwin application. The following metrics have been counted in the previous sections:

- $n(O_M) = 22$

  The cumulative complexity of objects equals 22.

- $n(P_M) = 26$

  The cumulative complexity of properties equals 26.

- $n(R_M) = 4$

  The cumulative complexity of relationships equals 4.

Inserting these values in Equation 7.1 we get the following cumulative method complexity:

$$C^{'}(M) = \sqrt{n(O_M)^2 + n(R_M)^2 + n(P_M)^2} = \sqrt{22^2 + 4^2 + 26^2} = 34,29 \tag{7.2}$$

7.5. Results

With the cumulative method complexity of 34.29, the process modeling component of the Darwin prototype can classified into the cumulative complexity table referenced in Chapter 3. This results in the ranking shown in Table 7.4.

Compared to the cumulative complexity of 48.18 of CMMN 1.0 calculated by Marin et al. [22], Darwins process modeler reduces the complexity by around 29%. This shows the reduction of complexity for the chosen subset and indicates that as

Method	**Objects**	**Relationships**	**Properties**	**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 1.4 Activity Diagrams	8	5	6	11.18

Table 7.4.: The cumulative complexity of Darwin compared to other process modeling notations. Author's own compilation based on Marin et al., 2014 [22]

a result, the process modeling language will be less hard for users to get familiar with. The fact that the subset of CMMN used in Darwins process modeler is still more complex than UML Activity Diagrams and Event-driven Process Chains can be attributed to the simplicity of the two notations.

While the complexity reduction is the most important property to be observed, it is still important to emphasize the close relation between the two specifications. This close relation is illustrated in the three-dimensional plot in Figure 7.1, showing the same trend for the subset that can be observed for the three subsets created for BPMN 1.2.

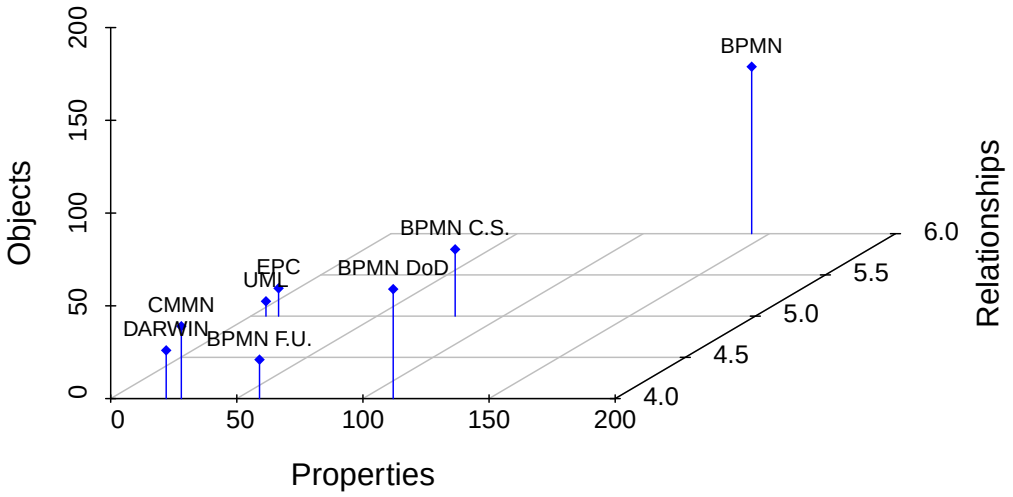


Figure 7.1.: The cumulative complexity of Darwin compared to other process modeling notations in an Object-Relationship-Property cube. Author's own compilation based on Marin et al., 2014 [22].

Part III.

Implementation

8. Research Prototype

The developed application is not standalone software, but serves as a module of an existing prototype. The existing prototype, which was developed at the Chair of Software Engineering for Business Information Systems at the Technical University of Munich, is an extended wiki system named Darwin, which aims to support Adaptive Case Management (cf. [13]). Users can create new cases and add pages and types which are linked to it.

Data is added to a specific case in the form of unstructured as well as structured information. Structured information can be added by creating new tasks and attributes, which can be linked to either a task or a page. By making use of a role management system, roles that are linked to a specific case get instant feedback on the tasks and workitems that need to be completed. This way, different users can work on cases in a collaborative fashion enabling an emergent design of processes.

The interactive CMMN process modeler is integrated into this application, to allow process modelers to create dependencies and constraints between the different activities of a case. The following sections describe the architecture, which the process modeler is built on.

8.1. Technical Architecture

The main objective of this section is to give a detailed overview of the technologies which were used in order to create the current prototype. As the prototype was created as part of a web-application, there are many solutions available which are capable of successfully implementing the requirements derived in the previous part of this work. The reason for this is mainly the current popularity of web-applications. This is especially true for applications which involve some sort of distributed collaboration. This section describes the used technologies in detail and tries to illustrate the advantages of it over another similar solution.

The Play! Framework[1], which is used for the implementation of Darwin, is a full-stack web framework using the Java Virtual Machine. According to [31] it is especially appealing to developers due to its use of state-of-the-art technology and useful features such as hot reloading of source files. While the first version of the framework only supported plain Java code, the second version has a strong focus on the Scala programming language, which combines functional and object-oriented programming. With regard to the implemented prototype, it is an ideal framework, as it comes with a built-in Model-View-Controller (MVC) architecture and provides a native REST (Representational State Transfer) interface. The latter enables the communication between backend and frontend. Resources provided by the backend can be accessed, modified, and deleted using simple URIs and HTTP-Requests (mainly GET and POST).

The persistence layer of Darwin is based on a MongoDB database[2], which is a NoSQL-database using Document-Oriented Storage to store its data. In order for Scala to support MongoDB, the Casbah interface[3] is used to provide flexible access. In order to use Object-relational mapping (ORM) for the objects created in Scala, the Salat plugin[4] for Play! 2 was used, which provides a serialization library for case classes.

In order to implement the required CMMN process editor in a web application, a library which allows the creation of custom shapes and diagrams is necessary. The open-source library JointJS[5] uses modern web technologies such as HTML 5 and Scalable Vector Graphics (SVG). JointJS was used in combination with the AngularJS framework[6], which is utilized throughout the entire Darwin application to extend the functionality of HTML, providing more dynamic functionality for web-applications. In the context of the process editor AngularJS provides the relevant lifecycle data, while JointJS uses this data to provide a graphical and dynamic user-interface to visualize the process using the CMMN-specific notation.

[1]projects website available at: https://www.playframework.com/, last accessed on 2014-12-03
[2]projects website available at: http://www.mongodb.org/, last accessed on 2014-12-03
[3]GitHub project available at: https://github.com/mongodb/casbah, last accessed on: 2014-12-03
[4]GitHub project available at: https://github.com/leon/play-salat, last accessed on: 2014-12-03
[5]projects website available at: http://www.jointjs.com/, last accessed on 2014-12-03
[6]projects website available at: https://angularjs.org/, last accessed on 2014-12-03

8.2. System Design

While the previous section focused mainly on the technologies, which were used for the implementation of the web-application, this section describes the design choices made. Since the implementational part of this thesis is concerned with the CMMN process editor, the main focus will be on its specific structure to provide a complete overview of the developed application.

8.2.1. Single Page Application

Even though the Darwin application was built using multiple pages, the CMMN module developed is following the Single Page Application (SPA) paradigm. An SPA is a web-application which is different to the *Client-Server* architecture that was formerly used by the majority of web-applications.

In a client-server architecture the client makes a *Request* for a resource and gets a *Response* from the server. The client uses this response to render the data according to the specific content of it. In most cases, this is done by a web-browser on the client side. The communication happens strictly synchronous, which means that there is no exchange of data between each *Request-Response* set.

More recently, the use of single page applications has grown rapidly, which can be attributed to new technologies that allow the old-fashioned client-server architecture to be extended to a more dynamic system. This new paradigm allows the web-application to communicate asynchronously with the server to load only specific parts of a page. This enables the web-browser to stay on a single page without reloading the whole page, whenever an exchange of data happens between the client and the server. The browser can now update the specific parts of the page which have changes, while leaving the rest unchanged. This does not only reduce the traffic between the two entities, but also enhances the user-experience for web-applications which react to changes in real-time.

In order to implement an SPA, the logic contained on the client side needs to be increased. This is mainly due to the fact that the server answers with specific responses containing only the necessary information, as opposed to sending a complete webpage which only needs to be rendered by the client. The use of JavaScript libraries like AngularJS and JointJS provides the foundation for this logic-driven frontend.

8.2.2. Interactive Frontend

The interactive frontend used to implement the process editor makes use of the aforementioned JavaScript libraries AngularJS and JointJS. In the context of this application, AngularJS provides the initial lifecycle data of the current process through a directive. The specific directive used, is illustrated in Listing 8.1. It shows how the new directive *ngPageCmmn* is connected to the *processdata*, which contains the relevant information for the CMMN process to be rendered in JointJS.

The creation of a new JavaScript prototype class is handled in line 10 of the code listing. A new *Process* class is created, using the information provided by the *processdata* element, which is formatted using a JSON object. Finally, the directive calls the *initPageLifecycle()* method, which triggers the drawing of the CMMN process model.

```
1  myDirectives.directive('ngPageCmmn', function() {
2    return {
3      replace : true,
4      link : function(scope, element, attrs) {
5        attrs.$observe('ngPageCmmn', function(processdata) {
6          if (processdata) {
7            var p = JSON.parse(processdata);
8            var selector = angular.element(element).attr('id');
9            angular.element(element).empty();
10                     var process = new Process(p.page, p.process,
                           selector, scope.loadPageData);
11                     pageProcess = process;
12                     process.initPageLifecycle(p.page);
13         }
14       });
15     }
16   };
17 });
```

Listing 8.1: Angular directive to provide lifecycle data for the Process prototype

Within the *Process* prototype it is now possible to create the available objects using JointJS. Further actions by users are propagated to the backend using the available REST interface. This way, the frontend is able to provide instant feedback to the user.

An example for this is the creation of a new task. When a modeling user specifies

the name and type of the new entity, an AJAX call is made to the backend containing the relevant information. Once the new task has been created in the backend and persisted in the database, a response is sent back to the frontend. Provided that the response was valid, a new task is visualized in the CMMN model. This enables front- and backend to maintain a consistent state, which is especially important for a collaborative application.

9. Implementation of Requirements

This chapter focuses on the documentation of the steps taken throughout the implementation of the requirements defined in Part II of this thesis. Figure 9.1 shows an image of the CMMN modeler implemented in the Darwin application. It shows the tight integration of the modeler and the rest of the prototype using the aforementioned SPA approach. The specific details of the different feature implemented are described in detail below.

The chapter starts with a description of the basic functionality implemented to provide an interactive interface for modelers to make use of CMMN. It is followed by a detailed outline of the different CMMN-specific requirements which have been implemented.

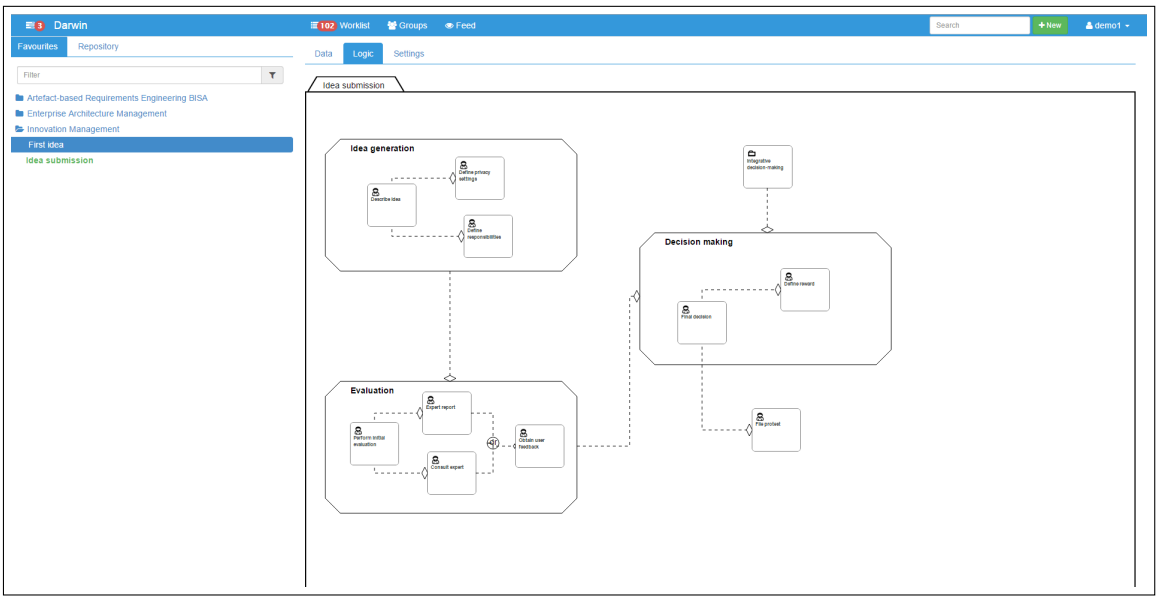


Figure 9.1.: View of the CMMN modeler integrated in the Darwin prototype.

9.1. Basic Functionality

In order to be able to model processes using the CMMN modeler, users need to be able to interactively create and structure a process. In the implemented solution users have a number of operations at hand, which lets them influence the logic and the visual representation of a process. An example of such operations is illustrated in Figure 9.2, which appear whenever the user hovers over the main process window.

Figure 9.2.: Case operations available to the Modeler role in the implemented prototype.

Generally, the operations available to a specific user are influenced by the role management system. A user with *Admin* rights will see all tools, while the *Modeler* role is only granted access to specific features. The general *Reader* role only provides read access and allows no process alteration.

As the structuring of a process relies heavily on the visualization, users can move stages and tasks in order to restructure the process in an appropriate manner. The current location of elements is propagated to the backend and stored in the database, which assures a uniform model throughout all clients.

9.2. CMMN Functionality

Before the implementation of the CMMN modeler, the process in Darwin was illustrated using simple elements to describe the different stages within a process. While it enabled users to get instant feedback of structure of a process, it did not make use of any concrete modeling techniques and CMMN-specific elements.

This section describes the different features which have been implemented throughout the course of this thesis based on the requirements extracted throughout the conceptual part.

9.2.1. Creation of Stages and Tasks

The two most basic structural elements in CMMN are Tasks and Stages. As mentioned before, they enable modelers to create hierarchical structures between the different activities of a case. Thus, the two concepts were implemented identical to their specification in CMMN 1.0.

Using the process modeling operations described above, the user can create a stage using the *Create Local Stage* action. In a modal window, the user has to enter the name of the stage. Once specified, the new stage appears in the process window. An example of a new stage is shown in Figure 9.3. The figure also shows the stage operations, which become available to modelers when hovering over it. As stages are used for creating hierarchical structures, they can be connected to other stages in sequence on the one hand and contain tasks on the other.

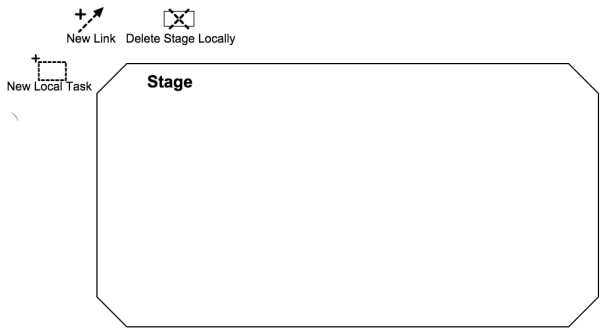


Figure 9.3.: New stage created in the CMMN process modeler, showing the available stage operations.

The creation of tasks works similar to the creation of stages. An additional input from the modeler is required, which specifies if the task is a *HumanTask* or *CaseTask*. This process is shown in Figure 9.5, which results in the creation of a new human task.

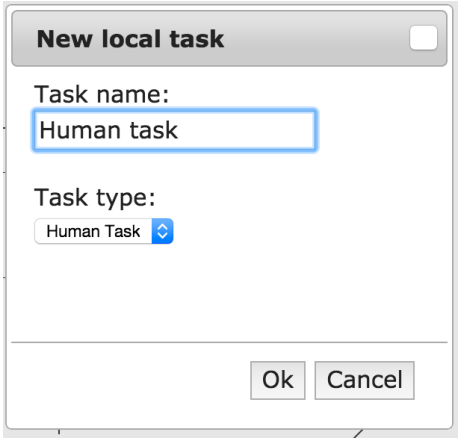


Figure 9.4.: Modal window to specify the name and type of a task.

When hovering over a task, the user can create dependencies between the current task and other stages and tasks. This state is depicted in Figure **??** which shows the newly created human task including the available operation icon. The following section describes the creation of dependencies in a more detailed way.

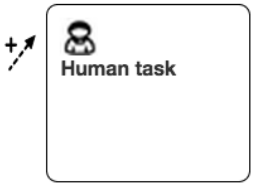


Figure 9.5.: Visual representation of a new HumanTask in the CMMN process modeler.

9.2.2. Creation of Dependencies

While Tasks and Stages enable modelers to create hierarchies, the creation of dependencies in CMMN is handled using Connectors. There exists only one representation of a connector in the specification which links any two elements of a case. The dotted line shown in Figure 9.6 indicates the existence of a producer-consumer dependency between the two tasks shown.

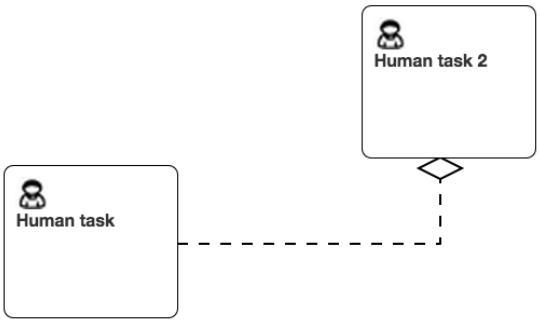


Figure 9.6.: Creation of a sequential order between two tasks using a connector.

The same connector is also used to model the AND and OR dependencies which are part of the implementation. In order to make these dependencies distinguishable from normal connections, a small circle collecting all incoming connections specifies the type of the constraint.

9.2.3. Cycle Prevention

As already mentioned before, the creation of cycles using connectors can lead to dead-locks and unpredictable states. For that reason, they are disallowed in the implemented process modeler. In order to enforce this requirement throughout the entire application, each creation of a dependency between two tasks needs to be checked for potential cycles.

In the implemented prototype, this is handled by the recursive *isTaskLinkCyclic()* function shown in Listing 9.1. Initially, it checks if there is a direct link pointing in the opposite direction between the source and target task. If this is not the case, the recursive function checks all possible connections that are connected to either the source or the target, if a cycle is present.

```
1 /** Checks if the taskLink would result in a cycle.   */
2   def isTaskLinkCyclic(sourceId: TaskId, targetId: TaskId, linkList:
        List[PageTaskLink]): Boolean = {
3     // Check if direct link between the two tasks exists
4     if (!linkList.filter(l => l.sourceId == targetId && l.targetId ==
          sourceId).isEmpty) return true
5     else {
6       val filteredList = linkList.filter(l => l.targetId == sourceId)
7       if (filteredList.isEmpty) return false
```

```
8       else {
9         var cycleDetected = false
10        for (link <- filteredList) {
11          // check recursively for all "relatives" if the new link
                results in a cycle
12          if (isTaskLinkCyclic(link.sourceId, targetId, linkList.
                filterNot(l => l.targetId == link.targetId)))
                cycleDetected = true
13        }
14        return cycleDetected
15      }
16    }
17  }
```

Listing 9.1: Scala method for preventing cycles

9.2.4. Progress Propagation

The constraints added to a case model using the CMMN modeler have a direct impact on the logic of the underlying process. If, for example, a Producer-Consumer dependency between two tasks is created, the consuming task needs to wait for its predecessor to finish. Such dependencies need to be propagated to the specific modules of the prototype handling the order of execution and the user-interface.

Due to the introduction of more complex constraints (*AND* and *OR*) the progress propagation is required to consider much more advanced states of a process. The *checkIfTaskIsFinished()* function, which usually just checked its predecessor for completion, needs to be able to observe the existence of such complex dependencies.

10. Evaluation

In order to evaluate the modeling capabilities of the implemented solution, the modeling of an actual process using the new features was also part of this thesis. It is based on the existing Innovation Management process at DATEV eG. The following sections introduce the underlying process and describe the usage of the implemented solution in order to model it in Darwin and create the logical dependencies and constraints.

10.1. The Innovation Management Process

The DATEV eG is one of Europe's largest software companies with almost 7000 employees [1] Their main focus is on providing services for tax, accounting and attorneys. Due to their constant need for new ideas, the company has developed an internal Innovation Management process, which aims at fueling the generation of innovation.

The company's innovation management process contains many logical dependencies between the different tasks that are part of it. Nevertheless, two independent solutions are used due to the evolving of the process over an extended period of time. One of these solutions is INITIATIV, a tool for the submission of ideas, which follows a strict pre-defined process and has been used for more than a decade. In contrast to this traditional solution, the DATEV Idea Pool (DIP) was developed in 2012 as an open innovation platform to foster the creation of new ideas and to promote collaboration. Even though DIP has many advantages over INITIATIV, some of the core processes are not implemented in the new solution, which creates some disadvantages over the traditional application. Due to the fact that both solutions complement each other, they are being used in parallel creating the need for a system combining both approaches.

[1] Based on the business year 2013. Information available at http://www.datev.de/, last accessed on 2014-12-06.

Since both tools show certain traits which are also part of the CMMN process modeler developed throughout the course of this thesis, it makes sense to look at the innovation management process employed at DATEV to analyze whether it can be modeled using the available features. On the one hand Darwin provides flexibility comparable to the collaborative DIP application, while on the other hand the internal process modeler allows for the creation of the necessary constraints which are mostly part of INITIATIV.

In order to be able to model the innovation management process in Darwin, the process needs to be explained. The following is a summary of the most important stages and the underlying tasks that are part of it.

Idea Generation

The initial *Idea Generation* stage mainly focuses on the creation of a new idea, which requires the innovative user to describe his idea and add the required attributes, presumably along with some more optional ones. Other tasks that are part of the idea generation process are the specification of the responsible department, which should be associated with the new idea, and the definition of the privacy settings, that define rules like access rights. In summary, the following three tasks are part of this stage: *Describe idea*, *Define responsible department*, and *Define privacy settings*. Additionally, there is a dependency between the *Describe idea* task and the other two. It is only possible to define a department and the responsibilities once an idea has actually been created and added to the workflow. For that reason, both definition tasks are consumers of the *Describe idea* task and should only be available to the end-users upon its completion.

Evaluation

The objective of the second stage of the process is the evaluation of the ideas added throughout the first stage. In order to distinguish between useful and useless ideas, the first task to be performed during this stage is *Perform initial check*, which can be regarded as a first control mechanism that triggers the following tasks, should the idea be regarded as innovative. The following two tasks require the input of an expert familiar with the specific nature of the idea. This expert can either be the person actually working on the idea evaluation or an external expert which needs to be consulted. The tasks *Create expert's*

report and *Consult expert* represent these distinct tasks. Both are dependent on the initial check creating a multiple-consumer dependency, comparable to the one in the first stage.

Finally, after the creation of an expert report, there is an option to *Obtain user feedback*. This task is dependent on the existence of an expert's report. For that reason, one of the two tasks involving experts needs to be completed - at least. A suitable constraint to be used for the creation of this dependency is an OR-Join.

Decision Making

The final stage of the innovation management process focuses on decision making. Here the ideas that have been evaluated in the second stage are ranked in order to make a decision on the best ideas. These actions are included in the *Make and explain final decision* task. As rewards are sometimes given to submitters of innovative ideas, the *Define reward for submitter* task is used to specify the relevant information.

There are also two important attributes of this stage which make it different to the other two. As there are various entities involved in the decision making process, there is also an option for users to file a protest (*File protest*). Nevertheless, this task cannot be attributed directly to the decision-making stage and needs to be placed outside of its borders, while still being dependent on the final decision task as a producer. Comparable to this situation, there is the option to extend the decision making process using the *Integrative decision making* approach, which requires to be connected to the whole stage referring to another case.

The three stages, along with the two tasks that are not part of any stage, describe the complete innovation management process, including its various dependencies. It is now possible to model the process using the features provided by Darwins process modeler, which is thoroughly explained in the following section.

10.2. Modeling of the Process

For the evaluation the innovation management process is modeled in this section. The main focus is on the creation of the existing dependencies and constraints between the different tasks that are part of the process.

In order to persist the innovation management process in Darwin, a new model needs to be created along with a new type (*Idea submission*). Using this type, pages can be created for each idea that refer to this type. Figure 10.1 shows what this initial setup looks like in the web frontend of Darwin. Along with the type, a sample page *First idea* has been created, which can both be seen in the panel to the left. The content window shows the description of the model which can be filled with context specific data.

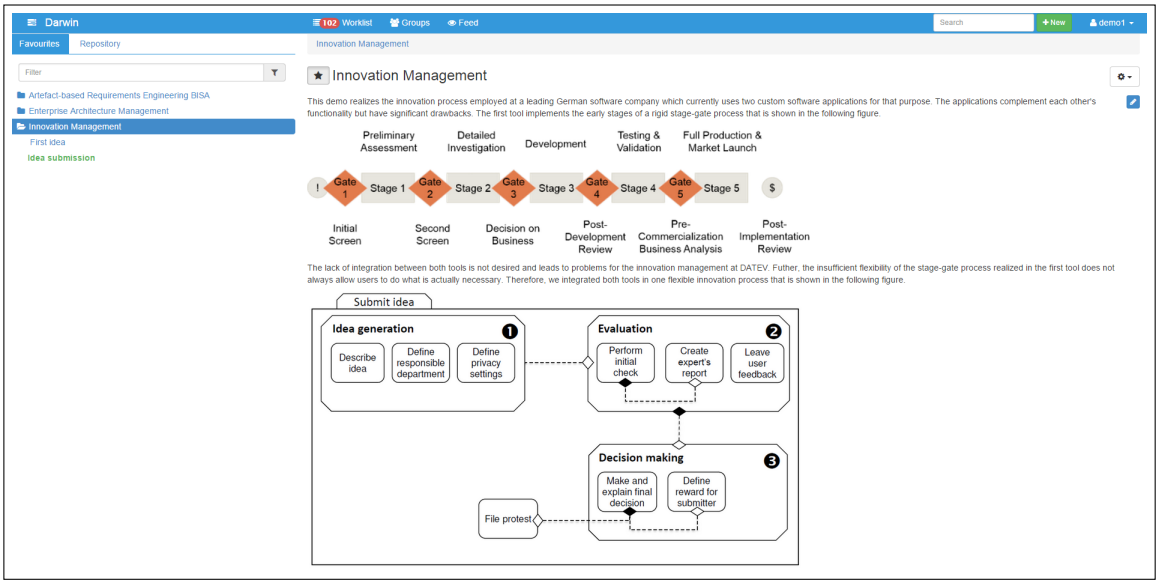


Figure 10.1.: The initial creation of the innovation management process in Darwin, showing model, type and page.

With the basic structure of the innovation management process defined, it is now possible to use the CMMN modeler in Darwin to describe the logic of the process. The three different stages of the innovation management process explained above represent an ideal structure to be modeled as different stages in the context of the process modeler.

In order to be able to modify the process, the *Modeler* role is required. Using the tools available to this role, the three stages can be created. This step is depicted

in Figure 10.2, showing the stages in the process modeler. Also shown are the Producer-Consumer dependencies between the stages, which were added using the stage-linking tool.

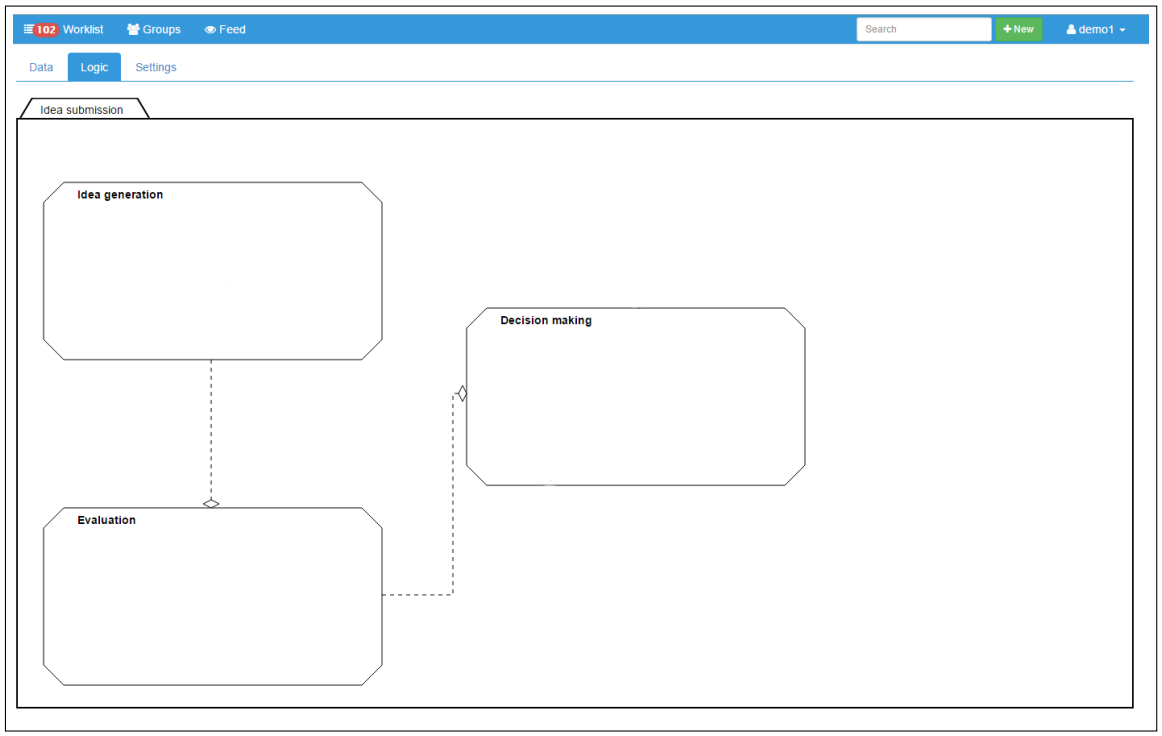


Figure 10.2.: The three stages of the innovation management process added to the CMMN model.

With the three necessary stages now modeled and linked, the necessary tasks that are part of each stage can be added. The same goes for the two tasks which do not have a stage and are added as tasks belonging to the whole case instead.

The next step is to create the numerous dependencies between the different tasks of the process. Using the task-linking tool, the sequential dependencies of the *Idea generation* and *Decision-making* stages can be created using the *Normal link* connector.

The connection of the CaseTask *Integrative Decision-making* with the *Decision-making* stage can be created the same way, as stages are shown in the dropdown menu for the target selection as well.

A more complex constraint is the OR-Join, which is part of the tasks in the *Evaluation* stage. Just like a normal connector between tasks, the AND and OR constraints can be added using the task-linking tool. Each of the two *expert* tasks is linked to the target the same way, by selecting the *OR-constraint* as link type.

In order to indicate the creation of an OR-Join, a small bubble is added, joining both incoming connectors. The resulting CMMN process is depicted in Figure 10.4, showing the three stages, all available tasks and the different constraints added to them.

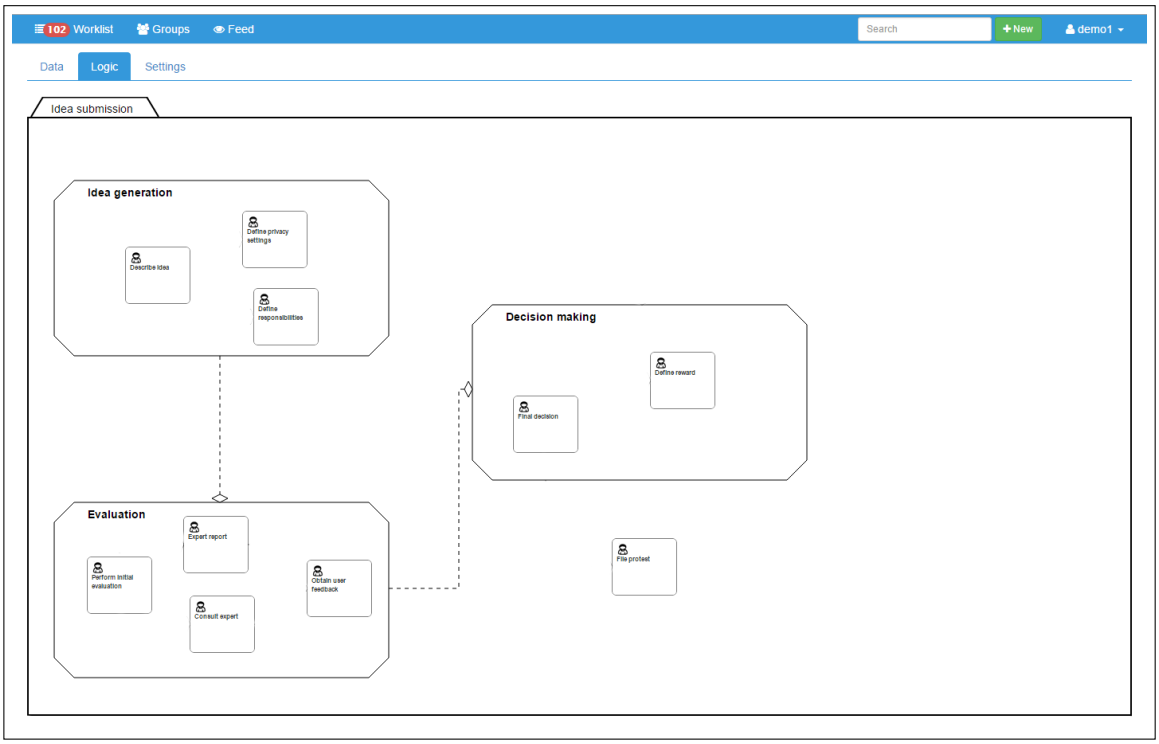


Figure 10.3.: HumanTasks and CaseTask added to the three stages and the case.

With the process now modeled in CMMN, end-users, adding information in the form of data to the Innovation Management case, will be influenced by the new constraints instantly. Depending on the currently active stage, only the tasks that do not have an unfinished predecessor appear in the worklist. This is illustrated in Figure 10.5, which shows the different stages of a worklist. When the user finishes on of the two tasks shown in *(a)* the subsequent task appears in the worklist as shown in *(b)*.

In summary, the modeling of the Innovation Management process at DATEV illustrated some of the key features provided by Darwins interactive CMMN modeler. Prior to the implementation of the modeler, only sequential processes could be modeled, making the structuring of a Knowledge-intensive Process like the one at hand impossible. By enabling users to create complex logical dependencies and constraints, the Innovation Management process can be modeled including some of

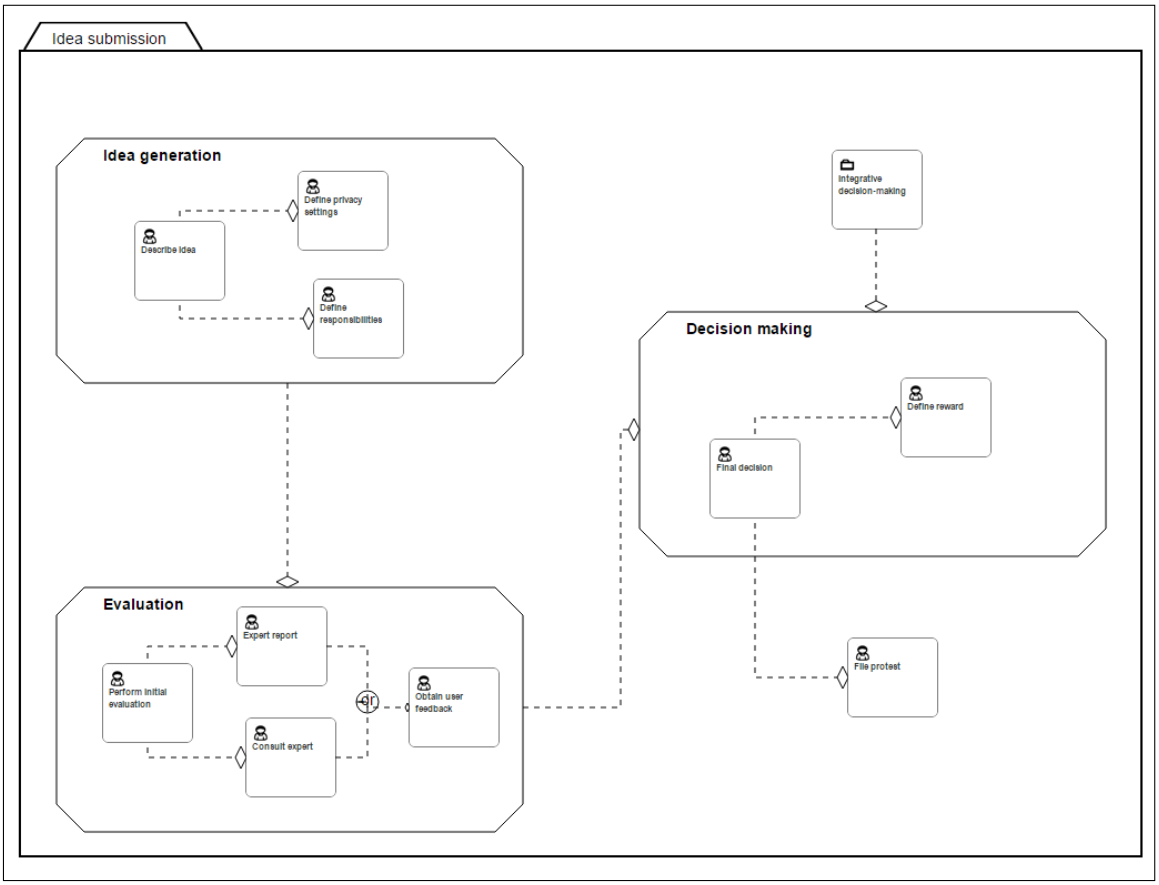


Figure 10.4.: The complete Innovation Management process modeled using Darwins CMMN editor.

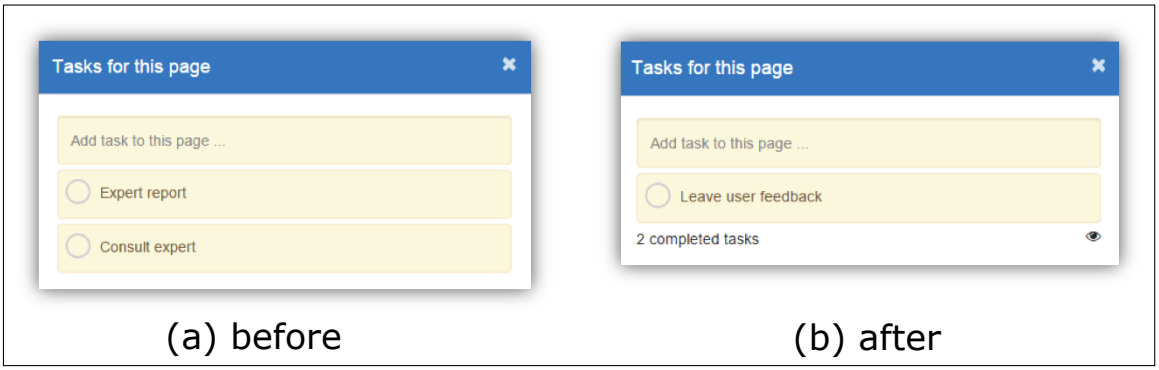


Figure 10.5.: The process logic directly influences the worklist of end-users.

the more advanced relations between the different activities.

Furthermore, end-users will be directly affected by the changes made on the process level. As they go through the Innovation Management process by adding

new information, the order of execution strictly follows the pattern created by the modeler. This enables modelers to control the execution of a process on the one hand, and to extract patterns, as the end-users keep on adding information in the form of tasks and attributes, on the other.

Part IV.

Conclusion and Outlook

11. Conclusion

This thesis focused on the thorough analysis of the Case Management Model and Notation and Knowledge-intensive Processes. It evaluated the ways in which the specification is able to support such processes and how it can be used in an actual software environment. In order to explain the relevant topics, they were discussed in detail throughout the first part of the thesis. Applying this relevant information, the second part of the thesis analyzed the requirements for the software prototype developed in the third part of this thesis. The developed prototype was then evaluated using an existing Knowledge-intensive Process.

Throughout the course of the thesis some interesting and important insights were gained which can be employed to further research performed in the area of Case Management. Using the three research questions introduced in the first chapter, the findings can now be compared to the objectives they specify.

Research Question 1: How can CMMN support users with the definition of Knowledge-intensive Processes?

The detailed introduction of the CMMN specification as well as the thorough analysis of the research performed by important experts in the area has shown that the modeling notation factors in the most important attributes of Knowledge-intensive Processes. It outlined some important events that ultimately resulted in the release of a final version, which leaves it up developers and process modelers to make use of the specification and render it usable in a software environment.

Furthermore, the complexity measurement performed for CMMN showed that the complexity can indeed be significantly reduced compared to BPMN 1.2 and its various subsets. While this serves as good evidence for the reduction of complexity achieved by CMMN, it should only be used as an indicator which can be used as the driver for performing further research.

It is also important to consider the fact that CMMN 1.0 has just recently been released in May, 2014. While it is definitely possible to draw first conclusions on its applicability for Knowledge-intensive Processes, it remains to be seen whether

the specification is capable of supporting them in actual business cases. Nevertheless, the two following research questions emblaze this issue in a more detailed way.

Research Question 2: What is a suitable subset of CMMN that can be used for Knowledge-intensive Processes?

To begin with, the complexity measurement performed for the subset used in Darwin showed that the complexity can indeed be significantly reduced compared to CMMN 1.0 as well as BPMN 1.2 and its various subsets. The creation of subsets, like the one used in the developed prototype, provide ways to reduce the complexity even further. While the creation of a subset to reduce complexity can generally be regarded as a reasonable approach, the extraction needs to be carefully executed in order to remain compliant with the CMMN specification.

In order to generate this subset, workflow patterns, which were described by the Workflow Management Coalition and further categorized by van der Aalst et al. at the Workflow Patterns Initiative[1], have been evaluated. In combination with the general requirements of Knowledge-intensive Processes and the CMMN specification, the workflow patterns which were found suitable for the implementation have been turned into dependencies and constraints to be used in the developed software environment.

Research Question 3: What is a suitable software environment for CMMN?

Using the identified subset of CMMN, the analyzed requirements were implemented in an existing research prototype. In order to be able to evaluate the implemented solution, an actual process used at a leading software company was modeled using the features provided by the software environment. It showed that it is indeed possible to create the necessary dependencies and constraints required by a complex business process, eliminating some of the major problems other solutions faced implementing the same model.

With regard to the great variety of Knowledge-intensive Processes, it is possible to define processes providing different amounts of flexibility. Should a process require only little guidance on the one hand, modelers can choose to not create any dependencies. However, if a process has certain attributes that require the creation of dependencies on the other hand, this can also be achieved using the constraints provided by the application. It enables modelers to create a broad range of different

[1]Website of the initiative: http://www.workflowpatterns.com, last accessed on 2014-12-03

processes, making the software system highly dynamic.

In summary, this thesis indicates that it is indeed possible to use the Case Management Model and Notation for supporting and structuring Knowledge-intensive Processes in various environments. Due to the high amount of flexibility provided to end-users as well as modeling experts, a broad array of different processes can be supported.

12. Future Outlook

Due to the fact that the Case Management Model and Notation, and its approach applied for handling Knowledge-intensive Processes, are still in their infancy, it is likely that new insights will lead to new developments. Many of the findings presented in this thesis leave room for further research and can be used as a starting point.

A logical next step concerning the developed CMMN process modeler would be to perform multiple case studies, in order to evaluate whether all processes can be modeled adequately in the application without missing dependencies. Should the need for more dependencies and constraints arise, the workflow patterns can be analyzed further in order to implement the new features in a suitable way. Due to the high flexibility required by Knowledge-intensive Processes, the implemented constraints in this prototype represent rather basic workflow patterns. It is therefore necessary to consider the issue of flexibility whenever new constraints are being added.

Another area of research could be the definition and implementation of a learning algorithm, in order to help modelers extract patterns from the information added to a case by end-users. In the current prototype, the modeler needs to find and extract patterns manually without any automation or guidance. Due to the constant adaption of Knowledge-intensive Processes, learning algorithms can use the environment already available in order to gain important insights and assist modelers to constantly improve the process.

A similar approach, which focuses more on the regular Knowledge-workers, would be the implementation of a recommendation system based on insights gained from similar processes. Such a system would be able to use information (e.g. naming of tasks and attributes) from prior executions of a process to give recommendations to users on-the-fly. This could enhance the user experience substantially and ultimately make Knowledge-workers more productive.

A more advanced area of research could explore methods to reduce complexity of process models which have been constantly improved. At some stage, processes

might become so complex that some dependencies turn into an obstruction. Once specified, it becomes increasingly harder to detect new patterns that require the removal of certain dependencies. While this issue could also be addressed by an algorithm, it requires more research to be performed.

All in all, case management for Knowledge-intensive Processes with the help of CMMN remains an interesting and novel field of research. Much of the research performed throughout the course of this thesis was based on literature which also mentioned this novelty. It will be exciting to track the further development of the CMMN specification and its use in real-life business cases. It is very likely that it will take some more time for it to move from a theoretical towards a practical solution.

Bibliography

[1] Dan Atwood. BPM process patterns: Repeatable design for BPM process models. *BP Trends May*, 2006.

[2] Manfred Broy. Requirements engineering as a key to holistic software quality. In *Computer and Information Sciences–ISCIS 2006*, pages 24–34. Springer, 2006.

[3] T.H. Davenport and L. Prusak. *Working Knowledge: How Organizations Manage what They Know*. Number 247 in EBSCO eBook Collection. Harvard Business School Press, 1998.

[4] Henk de Man. Case management: Cordys approach. *BPTrends, February*, pages 1–13, 2009.

[5] Claudio Di Ciccio, Andrea Marrella, and Alessandro Russo. Knowledge-intensive processes: An overview of contemporary approaches. *Knowledge-intensive Business Processes*, page 33, 2012.

[6] Claudio Di Ciccio, Andrea Marrella, and Alessandro Russo. Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches. *Journal on Data Semantics*, pages 1–29, 2014.

[7] Peter F Drucker. *The Age of Discontinuity: Guidelines to Our Changing Society*. Harper & Row, 1969.

[8] Peter F Drucker. Knowledge-worker productivity: The biggest challenge. *California Management Review*, 41(2):78–94, 1999.

[9] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: The issue of understandability. In *Enterprise, Business-Process and Information Systems Modeling*, pages 353–366. Springer, 2009.

[10] Dirk Fahland, Jan Mendling, Hajo A Reijers, Barbara Weber, Matthias Weidlich, and Stefan Zugal. Declarative versus imperative process modeling languages: the issue of maintainability. In *Business Process Management Workshops*, pages 477–488. Springer, 2010.

[11] Alexandra Fortis and Florin Fortis. Workflow patterns in process modeling. *arXiv preprint arXiv:0903.0053*, 2009.

[12] Martina E. Greiner, Tilo Bhmann, and Helmut Krcmar. A strategy for knowledge management. *Journal of Knowledge Management*, 11(6):3–15, 2007.

[13] Matheus Hauder, Dominik Münch, Felix Michel, Alexej Utz, and Florian Matthes. Examining adaptive case management to support processes for enterprise architecture management. *Enterprise Distributed Object Computing Conference Workshops (EDOCW). IEEE*, 2014.

[14] Christian Herrmann and Matthias Kurz. Adaptive Case Management: Supporting Knowledge Intensive Processes with IT Systems. In *S-BPM ONE-Learning by Doing-Doing by Learning*, pages 80–97. Springer, 2011.

[15] Gerhard Hube. Beitrag zur Beschreibung und Analyse von Wissensarbeit. 2005.

[16] Richard Hull, Francois Llirbat, Eric Siman, Jianwen Su, Guozhu Dong, Bharat Kumar, and Gang Zhou. Declarative workflows that support easy modification and dynamic browsing. In *ACM SIGSOFT Software Engineering Notes*, volume 24, pages 69–78. ACM, 1999.

[17] Marta Indulska, Michael zur Muehlen, and Jan Recker. Measuring Method Complexity: The Case of the Business Process Modeling Notation. *BPMcenter. org2009*, 2009.

[18] Santhosh Kumaran, Prabir Nandi, Terry Heath, Kumar Bhaskaran, and Raja Das. ADoc-oriented programming. In *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*, pages 334–341. IEEE, 2003.

[19] Craig Le Clair and Connie Moore. Dynamic Case Management - An Old Idea Catches New Fire. *Forrester Research*, 2009.

[20] John W Lloyd. Declarative programming in Escher. Technical report, Technical Report CSTR-95-013, Department of Computer Science, University of Bristol, 1995.

[21] Mike Marin, Richard Hull, and Roman Vaculín. Data centric BPM and the emerging case management standard: A short survey. In *Business Process Management Workshops*, pages 24–30. Springer, 2013.

[22] Mike A. Marin, Hugo Lotriet, and John A. Van Der Poll. 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.

[23] Hamid Reza Motahari-Nezhad, Claudio Bartolini, Sven Graupner, and Susan Spence. Adaptive case management in the social enterprise. In *Service-Oriented Computing*, pages 550–557. Springer, 2012.

[24] Anil Nigam and Nathan S Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[25] I. Nonaka and H. Takeuchi. *The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.

[26] OMG - Object Management Group. *Case Management Model and Notation*, version 1.0 edition, May 2014. `http://www.omg.org/spec/CMMN/1.0/`.

[27] G. O'Regan. *A Brief History of Computing*. SpringerLink : Bücher. Springer, 2012.

[28] Martin Owen and Jog Raj. BPMN and business process management. *Introduction to the New Business Process Modeling Standard*, 2003.

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

[30] Maja Pesic. Constraint-based workflow management systems: shifting control to users. 2008.

[31] Andy Petrella. *Learning Play! framework 2*. Packt Publishing, 2013.

[32] M. Pfiffner and P. Stadelmann. *Wissen wirksam machen: Wie Kopfarbeiter produktiv werden*. EM - edition MALIK. Campus Verlag, 2012.

[33] M. Polanyi and A. Sen. *The Tacit Dimension*. University of Chicago Press, 2009.

[34] Jan C Recker, Michael zur Muehlen, Keng Siau, John Erickson, and Marta Indulska. Measuring method complexity: UML versus BPMN. Association for Information Systems, 2009.

[35] J. Rehäuser and H. Krcmar. *Wissensmanagement im Unternehmen*. Arbeitspapiere. Lehrstuhl für Wirtschaftsinformatik, Univ. Hohenheim, 1996.

[36] Matti Rossi and Sjaak Brinkkemper. Complexity metrics for systems development methods and techniques. *Information Systems*, 21(2):209–227, 1996.

[37] Nick Russell, Arthur H. M. Ter Hofstede, and Nataliya Mulyar. Workflow control-flow patterns: A revised view. Technical report, 2006.

[38] Keng Siau and Matti Rossi. Evaluation of information modeling methods - a review. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 5, pages 314–322. IEEE, 1998.

[39] Keith D Swenson. *Mastering the Unpredictable*. Meghan-Kiffer Press, 2010.

[40] Thanh Thi Kim Tran, Max J Pucher, Jan Mendling, and Christoph Ruhsam. Setup and Maintenance Factors of ACM Systems. In *On the Move to Meaningful Internet Systems: OTM 2013 Workshops*, pages 172–177. Springer, 2013.

[41] U.S. Department of Defense. *Enterprise Architecture based on Design Primitives and Patterns Guidelines for the Design and Development of Event-Trace Descriptions (DoDAF OV-6c) using BPMN*, 2009.

[42] Roman Vaculin, Richard Hull, Terry Heath, Craig Cochran, Anil Nigam, and Piyawadee Sukaviriya. 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.

[43] Wil MP van Der Aalst, Arthur HM Ter Hofstede, Bartek Kiepuszewski, and Alistair P Barros. Workflow patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.

[44] Wil MP Van der Aalst, Mathias Weske, and Dolf Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.

[45] WMP Van der Aalst, Moniek Stoffele, and JWF Wamelink. Case handling in construction. *Automation in Construction*, 12(3):303–320, 2003.

[46] R Hevner von Alan, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.

[47] Stephen A White. Process modeling notations and workflow patterns. *Workflow handbook*, 2004:265–294, 2004.

[48] The Workflow Management Coalition. *The Workflow Management Coalition Terminology & Glossary*, version 2.0 edition, 1999.

[49] Michael zur Muehlen and Danny T Ho. Service process innovation: a case study of BPMN in practice. In *Hawaii international conference on system sciences, proceedings of the 41st annual*, pages 372–372. IEEE, 2008.

[50] Michael zur Muehlen and Jan Recker. 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.

[51] Michael zur Muehlen, Jan Recker, and Marta Indulska. Sometimes less is more: are process modeling languages overly complex? In *EDOC Conference Workshop, 2007. EDOC'07. Eleventh International IEEE*, pages 197–204. IEEE, 2007.