

Adaptive Case Management

Leveraging open source technologies to perform knowledge work

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Information and Knowledge Management

eingereicht von

Georg Kotschy

Matrikelnummer 9702603

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: ao. Univ.-Prof. Dr. Jürgen Dorn

Wien, 23.10.2017

(Unterschrift Verfasser/in)

(Unterschrift Betreuer/in)

Erklärung zur Verfassung der Arbeit

Georg Kotschy
Gassnerweg 2
4820 Bad Ischl

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 23.10.2017

(Unterschrift Verfasser/in)

Kurzfassung

Die öffentliche Verwaltung befasst sich mit einer Vielzahl von Ermittlungs- und Fallbezogenen Aktivitäten. Der Fortschritt eines Falles wird oft von den Umständen oder Ereignissen bestimmt, die während des Falls selbst eintreten und die vom Wissen eines Teams von Fallarbeitern abhängen. Die Arbeit des Fallteams ist Wissensarbeit, und nicht-traditionelle Prozessautomatisierungstechniken sind erforderlich, um Fallarbeiter zu unterstützen. Basierend auf den Erfahrungen aus aktuellen Case-Management-Projekten der Europäischen Kommission ist das Ziel dieser Diplomarbeit, einen Case-Management-Ansatz zu identifizieren, der für den Bereich von Handelsabkommen geeignet ist, sowie einen Open-Source-basierten Technologiestack vorzuschlagen, der den gewählten Ansatz unterstützen kann.

Diese Diplomarbeit wird Entscheidungsträgern Argumente für die Annahme eines adaptiven Ansatzes für das Fallmanagement auf der Grundlage des von den Wissensarbeitern benötigten Flexibilitätsbedarfs zur Erreichung der für sie festgelegten Ziele vorgeben. Business Analysten, die derzeit weitgehend vom Business Process Model und Notation (BPMN) abhängen, um die Arbeitsweise in einer Organisation darzustellen, werden von den Erkenntnissen zur Geschäftsprozessausführung in IT-Systemen profitieren, während sich Architekten und Entwickler von die Software-Design-Muster und Komponenten, die für die in dieser Arbeit vorgestellte Anwendungsarchitektur ausgewählt wurden, um weitere Untersuchungen zur Ausführung von emergenten Prozessen durchzuführen.

Abstract

Public administrations deal with a wide range of investigative and case-related activities. The progress of a case is often driven by the circumstances or events that occur during the case itself and which depend on the knowledge of a team of case workers. The work performed by the case team is knowledge work, and non-traditional process automation techniques are required to support case workers. Based on the lessons learned from recent case management projects at the European Commission, the aim of this research is to identify a case management approach suitable for the domain of trade negotiations, as well as an open source-based technology stack able to support the chosen approach.

This thesis will present decision-makers with arguments in favour of adopting an adaptive approach to case management, based on the need for flexibility required by knowledge workers to meet the goals set out for them. Business analysts, which currently largely depend on the Business Process Model and Notation (BPMN) to depict the way work is performed in an organisation, will benefit from the insights presented on business process execution in IT systems, while architects and developers can draw inspiration from the software design patterns and components chosen for the application architecture presented in this thesis, to conduct further research on the execution of emergent processes.

Table of Contents

Erklärung zur Verfassung der Arbeit	2
Kurzfassung	3
Abstract	4
Table of Contents	5
Index of Figures	8
Index of Tables	9
1 Introduction	10
1.1 Problem Definition	10
1.2 Methodology.....	11
1.3 Approach	12
1.4 Expected Results.....	12
2 State of the Art	13
2.1 Open Source-based Technologies	13
2.2 Case-based Reasoning	14
2.3 Knowledge Work.....	14
2.4 Business Process Orientation.....	15
2.5 Knowledge-intensive Processes	16
2.6 Domains of Predictability	18
2.7 Case Management Approaches	19
2.7.1 Business Process Management.....	20
2.7.2 Production Case Management.....	20
2.7.3 Adaptive Case Management.....	20
2.7.4 Comparison of Case Management Approaches	21
2.7.4.1 Predictability	21
2.7.4.2 Case Volume	22
2.7.4.3 Domain Knowledge	22
2.7.4.4 Worker Skill.....	22
2.7.4.5 Modelling Notation	22
2.7.4.6 Execution Engine	23
2.8 Process and Case Modelling.....	23
2.8.1 Business Process Model and Notation	23

2.8.2	Case Management Model and Notation	25
3	Core Requirements	27
3.1	Elements of Adaptive Case Management.....	27
3.1.1	Lifecycle.....	28
3.1.2	Case Templates	29
3.1.3	Decision Making	30
3.1.4	Content	30
3.1.4.1	Inbound Content.....	30
3.1.4.2	Outbound Content.....	31
3.1.4.3	Record Management	31
3.1.5	Security.....	31
3.1.5.1	Authentication.....	31
3.1.5.2	Authorisation.....	32
3.1.5.3	Encryption.....	32
3.1.5.4	Confidentiality	32
3.1.5.5	Authenticity.....	33
3.1.6	Accountability	33
3.1.7	Transparency	33
3.2	Mapping CMMN to ACM.....	33
3.2.1	ACM Notation Requirements.....	34
3.2.2	Coverage in CMMN.....	36
3.3	Case Management Rationalisation	37
3.3.1	Lifecycle.....	38
3.3.1.1	Case Management Lifecycle.....	38
3.3.1.2	Document and Record Management Lifecycle.....	39
3.3.2	Activities	40
3.3.2.1	Case Management Activities	41
3.3.2.2	Document and Record Management Activities	42
3.3.3	Building Blocks.....	44
3.4	Requirements Selection for Prototype.....	45
3.4.1	Interoperability	46
3.4.2	Process Execution	47
3.4.3	Decision Making	47

3.4.4	Content Management	47
3.4.5	Identity and Access Management	48
3.4.6	Accountability and Transparency.....	48
4	Practical Application of Case Management	49
4.1	Stakeholders.....	50
4.1.1	External Stakeholders.....	50
4.1.2	Internal Stakeholders.....	52
4.2	Predictability.....	53
4.2.1	High Predictability	54
4.2.2	Low Predictability	54
5	Prototype.....	55
5.1	Web Application.....	56
5.1.1	Programming Language	56
5.1.2	Application Server.....	56
5.1.3	Application Framework.....	57
5.1.3.1	Spring Context	60
5.1.3.2	Spring MVC.....	61
5.1.3.3	Spring Webflow	62
5.2	Application Architecture	64
5.2.1	Security Layer	64
5.2.1.1	Spring Security.....	65
5.2.1.2	LDAP	66
5.2.2	Persistence Layer.....	69
5.2.3	Service Layer.....	71
5.2.3.1	AcmFacade	71
5.2.3.2	Business Activity Services (BAS)	73
5.2.3.3	Flow Services (FS).....	74
5.2.3.4	Validation Services (VS)	74
5.2.4	Process Layer	75
5.2.4.1	Process Services (PS).....	75
5.2.4.2	Activiti	75
5.2.5	Presentation Layer.....	77
6	Test Scenarios	78

6.1	Test Scenario 01: State-based Choice	79
6.2	Test Scenario 02: Signal-based Choice	80
6.3	Test Scenario 03: Repetitive Choice	81
6.4	Test Scenario 04: Run-time Choice.....	83
6.5	Test Scenario 05: Process Migration	84
6.6	Test Scenario 06: Task Templates.....	85
7	Evaluation.....	87
7.1	Interoperability	87
7.2	Process Execution.....	89
7.3	Decision Making.....	89
7.4	Content Management.....	90
7.5	Identity and Access Management.....	91
7.6	Accountability and Transparency.....	93
8	Conclusion.....	94
	Bibliography.....	97

Index of Figures

FIGURE 1	SCIENTIFIC METHOD (LEFT) VS ENGINEERING DESIGN PROCESS (RIGHT) [1]	11
FIGURE 2	CLASSIFICATION STRUCTURE FOR KNOWLEDGE-INTENSIVE PROCESSES [11]	16
FIGURE 3	DOMAINS OF PREDICTABILITY [17].....	18
FIGURE 4	BPMN 2.0 - BUSINESS PROCESS MODEL AND NOTATION [23].....	24
FIGURE 5	CASE MANAGEMENT MODEL AND NOTATION [25].....	26
FIGURE 6	BUSINESS PROCESS MANAGEMENT LIFECYCLE [28].....	28
FIGURE 7	CMR CASE MANAGEMENT LIFECYCLE	38
FIGURE 8	CMR DOCUMENT AND RECORD MANAGEMENT LIFECYCLE.....	40
FIGURE 9	EXTERNAL STAKEHOLDERS OF IT PROJECTS AT THE EC.....	51
FIGURE 10	PM ² PROJECT ORGANISATION [43]	52
FIGURE 11	EU REVENUE AND CUSTOMS DUTIES [44].....	54
FIGURE 12	STATE OF EU TRADE 2017 [47].....	55
FIGURE 13	TIOBE PROGRAMMING COMMUNITY INDEX [51]	56
FIGURE 14	JAVA APPLICATION SERVER MARKET DISTRIBUTION [55].....	57
FIGURE 15	WEB APPLICATION FRAMEWORK USAGE SINCE 2012 [57]	57
FIGURE 16	SPRING MVC DISPATCHERServlet [64].....	61
FIGURE 17	APPLICATION ARCHITECTURE OF ACM PROTOTYPE.....	64
FIGURE 18	ROLE OF ACMFAÇADE IN SERVICE LANDSCAPE.....	72
FIGURE 19	CREATE ATTACHMENT WEBFLOW	74
FIGURE 20	ACTIVITI PROCESS MODEL FOR STATE-BASED CHOICE	80
FIGURE 21	CONFIGURATION OF TASK FORM IN ACTIVITI DESIGNER	80
FIGURE 22	ACTIVITI PROCESS MODEL FOR SIGNAL-BASED CHOICE.....	81
FIGURE 23	CONFIGURATION OF PROCESS SIGNALS IN ACTIVITI DESIGNER	81

FIGURE 24 ACTIVITI PROCESS MODEL FOR REPETITIVE CHOICE	82
FIGURE 25 SCREENSHOT OF PROCESS INSTANCE UI SHOWING REPETITIVE CHOICE	82
FIGURE 26 ACTIVITI PROCESS MODEL FOR RUN-TIME CHOICE	83
FIGURE 27 SCREENSHOT OF PROCESS INSTANCE UI SHOWING SUBPROCESS CHOSEN AT RUN-TIME	84
FIGURE 28 ACTIVITI PROCESS MODEL FOR MIGRATION (VERSION 1)	85
FIGURE 29 ACTIVITI PROCESS MODEL FOR MIGRATION (VERSION 2)	85
FIGURE 30 SCREENSHOT OF PROCESS INSTANCE UI SHOWING MIGRATION.....	85
FIGURE 31 ACTIVITI PROCESS MODEL FOR TASK TEMPLATES.....	86
FIGURE 32 SCREENSHOT OF GROUP MANAGEMENT UI	88
FIGURE 33 SCREENSHOT OF CREATE ATTACHMENT UI	91
FIGURE 34 CONFIGURATION OF USER TASK IN ACTIVITI DESIGNER.....	92
FIGURE 35 SCREENSHOT OF PERSONALISED DASHBOARD.....	94
FIGURE 36 TRIPLE CROWN OF PROCESS IMPROVEMENT STANDARDS [88]	96

Index of Tables

TABLE 1 COMPARISON OF CASE MANAGEMENT APPROACHES.....	21
TABLE 2 CMR CASE MANAGEMENT ACTIVITIES	42
TABLE 3 CMR DOCUMENT AND RECORD MANAGEMENT ACTIVITIES	44
TABLE 4 MAPPING OF SELECTED ACM REQUIREMENTS	46
TABLE 5 DEPLOYMENT DESCRIPTOR FOR ACM PROTOTYPE.....	59
TABLE 6 EXCERPT FROM ROOT CONTEXT OF ACM PROTOTYPE	61
TABLE 7 EXCERPT FROM SERVLET CONTEXT OF ACM PROTOTYPE.....	62
TABLE 8 EXCERPT FROM SERVLET CONTEXT OF ACM PROTOTYPE.....	63
TABLE 9 EXCERPT FROM SECURITY CONTEXT OF ACM PROTOTYPE	66
TABLE 10 EXCERPT FROM ROOT CONTEXT OF ACM PROTOTYPE	68
TABLE 11 EXCERPT FROM AcMLDAPGROUPMANAGER SOURCE CODE	69
TABLE 12 EXCERPT FROM ROOT CONTEXT OF ACM PROTOTYPE	70
TABLE 13 EXCERPT FROM ROOT CONTEXT OF ACM PROTOTYPE	77
TABLE 14 EXCERPT FROM SERVLET CONTEXT OF ACM PROTOTYPE.....	78
TABLE 15 EXCERPT FROM TILES CONFIGURATION OF ACM PROTOTYPE	78
TABLE 16 EXCERPT FROM ACTIVITI TASK QUERY DEFINITION	87
TABLE 17 COLLABORATOR LIFECYCLE FOR PROCESS ENTITIES	93

1 Introduction

The European Commission (EC) deals with a wide range of investigative and case-related activities to enforce European Union (EU) law, e.g. state aid, anti-trust, cartels, mergers, customs, fishing quotas, anti-dumping, anti-subsidy, safeguards, trade negotiations and many more. The progress of a case is often driven by the circumstances or events that occur during the case itself and which depend on the knowledge of a team of case workers. The work performed by the case team is knowledge work, and non-traditional process automation techniques are needed to support case workers.

Several recent and ongoing projects at the EC aim to meet the needs of case workers involved in one or more of the mentioned fields: DG Trade developed the Negotiation Support Tool (NEST) prototype, assisting trade negotiators; DG Taxation and Customs Union will release Customs Decisions (CD) in 2017, connecting economic operators with customs officials; DG Competition is undertaking the Case Management Rationalisation (CMR) exercise, aiming to support a wide variety of case workers across several business domains.

In December 2013 the NEST project was put on hold after the completion of the NEST prototype. A post-mortem analysis showed that the prototype failed to meet trade negotiators need for flexibility, thus preventing them from achieving the goals set out for the negotiation team. Consequently, it was decided that further research was needed into case management approaches to identify an approach suitable for the emergent nature of trade negotiations.

This thesis focuses on the research conducted in the aftermath of the NEST project in 2014, while also relying on the results and lessons learned from two other case management projects undertaken by the EC. The CMR project serves as a reference source for case management requirements in public administration, while the CD project provides a counter-example to the open source-based approach of NEST, highlighting the limitations of vendor-based process execution solutions.

1.1 Problem Definition

The domain of trade negotiations has demonstrated the need for case management solutions capable of handling unpredictable processes. Based on the post-mortem analysis of NEST, the aim of this research is to identify a case management approach suitable for the domain of trade negotiations, as well as an open source-based technology stack able to support the chosen approach. The following three research questions will help define the scope of the problem under investigation.

- **RQ1:** Does a suitable case management approach for highly unpredictable business domains exist?
- **RQ2:** Can this case management approach be supported by an open source-based technology stack?
- **RQ3:** Can a business process execution engine be customised to allow for the execution of unpredictable processes?

The first question deals with the general approach chosen for case management. Based on the post-mortem analysis of the NEST prototype, the business process-oriented approach chosen for the NEST project does not sufficiently meet the requirements of trade negotiators for flexibility in planning the work necessary to meet the goals set out for them. The first area

of research therefore focuses on identifying an approach for managing emerging processes, i.e. a course of events unfolding step-by-step, with each step yielding knowledge that may influence the execution of the following step.

The second question asks whether such a case management system can be built from open source-based components. While vendor-based solutions attempt to meet the needs of a broad customer base, their proprietary nature prevents the customisation required for an innovative approach to case management. Open source-based software on the other hand allows project teams engaged in process execution research to create and maintain their own fork of the source code, and to customise it to meet specific needs set out by their project.

The third and final question results from the two previous questions, and focuses on the execution of emergent processes through customisation of an existing process execution engine. This question requires exploratory research to analyse the chosen process execution engine, and to identify possible approaches allowing for unpredictability in process execution.

1.2 Methodology

This thesis addresses an engineering problem, and applies the engineering design process to investigate open source-based execution of unpredictable processes.



Figure 1 Scientific method (left) vs engineering design process (right) [1]

As shown in the figure above, the scientific method (on the left-hand side) and the engineering design process (on the right-hand side) follow a similar pattern. However, unlike the scientific method which uses an experiment to prove a hypothesis, the engineering design process relies on prototyping and testing a proposed solution.

Having defined the problem, background research will serve to specify requirements and choose a solution approach. A prototype is designed and tested, and the test results are compared to the requirements set out for the solution. The conclusion from this comparison is then discussed to highlight areas of interest for further research.

1.3 Approach

Based on the problem definition in chapter 1.1, the background research presented in chapter 2 focuses on the nature of knowledge work and existing approaches to case management. This phase of the thesis was conducted in early 2014, and relies on a review of literature available at that time.

Chapter 3 establishes the core requirements of an adaptive case management (ACM) system for public administrations, through a detailed examination of: the elements of adaptive case management, as defined by Max Pucher in a series of contributions to Keith Swenson's books on supporting unpredictability in knowledge work; a mapping of the Case Management Model and Notation (CMMN) to ACM, as discussed in a publication by Kurz et al. in S-BPM ONE '15; and the functional requirements identified by the CMR project, which represent the needs of a wide variety of case workers across several business domains of the EC.

Chapter 4 discusses lessons learned from the original NEST prototype and the CD project, both of which followed a non-adaptive approach to case management. The experience and knowledge gained from these projects serve as the inspiration for the solution proposed in this thesis.

In chapter 5 the application architecture of the ACM prototype developed for this thesis is presented. Relying on an open source- and standards-based approach to software development, components suited to meet the requirements set forth in chapter 3.4 are identified, and their role in a case management system is demonstrated using code excerpts from the prototype.

The ACM prototype is tested through various modelling patterns, demonstrating the prototype's capabilities for executing unpredictable processes. The test scenarios and results are presented in chapter 6.

Chapter 7 compares the functionality of the ACM prototype against the core requirements discussed in chapter 3.4.

Finally, a conclusion of the research performed is communicated in chapter 8, pointing out areas recommended for further research.

1.4 Expected Results

The aim of this research is to investigate how open source-based technologies can support knowledge work in public administration. Decision-makers will be presented with arguments in favour of adopting an adaptive approach to case management, based on the need for flexibility required by knowledge workers to meet the goals set out for them. Business analysts, which currently largely depend on the Business Process Model and Notation (BPMN) to depict the way work is performed in an organisation, will benefit from the insights presented on business process execution in IT systems, while architects and developers can draw inspiration from the software design patterns and components chosen for the

application architecture presented in this thesis, to conduct further research on the execution of emergent processes.

2 State of the Art

The Negotiation Support Tool (NEST) prototype, developed by the Directorate-General for Trade (DG Trade) in 2013, followed a business process-oriented approach to supporting trade negotiations. The post-mortem analysis of the project concluded that traditional business process execution does not sufficiently meet the level of predictability (and thus need for flexibility) encountered in this business domain. While certain elements of the chosen approach (e.g. open source-based technology stack) seemed promising, research into the field of case management was required to better understand the unpredictability of knowledge work, and identify a case management approach better suited for unpredictable business domains such as trade negotiations.

This chapter examines the underlying concepts of case management systems based on a review of available literature. The background research for this thesis was conducted in early 2014, and depends largely on published books and articles by early adopters of Adaptive Case Management (ACM) in the software industry. ACM has since become an increasingly popular topic in the academic field, and a number of academic papers related to ACM have been published after 2014.

After starting with a short explanation of open source-based technologies and case-based reasoning, the nature of knowledge work (and how it differs from routine work) is discussed. This is followed by an introduction to business process-orientation, and a classification of the knowledge-intensive processes encountered in knowledge work. A review of the domains of predictability leads into a definition of case management, and a discussion on three popular approaches to case management. Finally, the two dominant standards for process and case modelling are presented.

2.1 Open Source-based Technologies

The available functionalities of an off-the-shelf product are based on compromises and assumptions, which may not hold true for every customer. Existing vendor lock-in is always a convincing argument for choosing vendor-based solutions, but open source technologies offer a viable alternative. A higher learning curve and implementation cost is offset by the ability to adapt and the absence of recurring licensing fees.

Vendor-based solutions hold the promise of scaling benefits and technical support, in return for a hefty licensing fee. While it makes absolutely no sense to use e.g. Oracle BPM alone as a process execution engine, once you add the other elements of the Oracle stack (i.e. database, enterprise service bus, web server, load balancer etc...), the synergies may lead to a result that is greater than the sum of its parts. This advantage comes at a price though, as there is no guarantee that each element of the stack features the same level of quality, yet the scaling benefits depend on implementing the entire stack.

Open source software on the other hand is free, and allows the development team full control over the software itself. This makes, what is effectively a “Sophie’s choice” of available features versus available budget in the context of vendor-based solutions, into a more intricate weighing of interdependent factors in the context of open source-based solutions. Wide acceptance of a solution reduces the risk that the community embraces forks or

reboots of a project, and increases its lifespan. A large developer community ensures that the project is regularly updated and bugs are fixed in a timely manner. Finally a solid architecture provides a strong foundation for customisation of the source code.

Another less intuitive benefit of the open source concept lies in dealing with large patent portfolios. Tech companies are increasingly pooling patents in open source projects, thus avoiding endless patent disputes, and allowing for collaboration between competitors. An example of this is Google's open source web-based user interface framework Angular, which is written in Microsoft's open source TypeScript, a programming language that compiles to JavaScript, which is a web browser-based scripting language originally created by Netscape. [2]

2.2 Case-based Reasoning

Case-based reasoning (CBR) is an artificial intelligence method that recommends solutions based on previously solved problems. When a new problem is encountered, the most similar historically known problem is identified, and its solution is adapted to the new problem. The new problem and the adapted solution are recorded, such that the problem/solution repository grows with every new problem. Two major factors influencing the success of CBR are the predictability and repeatability of work. [3]

Low predictability makes it difficult to identify similar problems that have been encountered before. As CBR matches a problem to a previously applied solution, a large set of potential problems has a detrimental effect on the success of the matching algorithm. If anything can happen during a case, it is impossible to determine the correct solution in advance. Instead, one has to wait for the case to unfold step-by-step, and whenever a problem becomes visible, choose the appropriate solution for that problem. This is the central issue that traditional business process management fails to address, and where the need for an adaptive approach to case management arises. Not being able to react, and to adapt a case to the changing circumstances of the world in which it unfolds, renders a case management system essentially useless, as case workers are forced to work outside of the system to solve their problem.

Low repeatability makes it unlikely that a solution will be re-used in the future. The complexity of a case depends largely on the decisions contained in it. More decisions (and a higher factor of outgoing transitions per decision) lead to a larger number of possible paths, i.e. distinct sequences of activities that lead from start to finish. As case based reasoning matches a problem to a previously applied solution, a large set of potential solutions has a detrimental effect on the success of the matching algorithm. Even if a specific case has a low complexity, it may still suffer from low repeatability due to its practical rate of occurrence. If the organisation managing the case is small, its overall caseload may not be large enough to see two occurrences of a certain case. If the probability of a certain case occurring is very low, even an organisation as large as the European Commission (EC) may never encounter it.

2.3 Knowledge Work

Routine work is predictable and repeatable. Due to its predictability, it can be planned in advance, i.e. a pattern can be identified before routine work is executed. While there may be slight differences in how routine work is performed from instance to instance, it is sufficiently

similar to allow identification of a pattern to be feasible. Its repeatability further allows routine work to be automated through the use of traditional process automation techniques. [4]

Knowledge work on the other hand is unpredictable, non-repeatable, emergent and goal-oriented. As the course of events may vary greatly from instance to instance, the specific sequence of human tasks involved in knowledge work cannot be planned in advance. Knowledge work is also rarely repeated the same way for several instances in a row, making identification of a pattern unfeasible. Given that the course of events unfolds step by step, and that every step can yield knowledge that may influence the selection of the following step, knowledge work is emergent, i.e. it is planned during execution rather than before execution. Since knowledge work cannot be defined as a series of transitions and gateways, it is instead defined as a set of goals to be fulfilled by the knowledge worker, a person who has a high degree of expertise, education or experience, and whose job focuses on the creation, distribution or application of knowledge. [4]

The nature of knowledge work requires IT solutions that empower knowledge workers with the tools required to achieve their goals, rather than limiting their flexibility to act appropriately by forcing them down a pre-determined path.

2.4 Business Process Orientation

“A concept that suggests that organizations could enhance their overall performance by viewing all the activities as linked together into a process that ultimately produces a good or service.” [5]

Business Process Management (BPM) is heavily inspired by Frederick Winslow Taylor's theory of scientific management. Taylor advocated the separation of planning and execution, requiring businesses to think first, then act. Business process models are effective means of planning and structuring predictable work before execution (i.e. at design-time). [6] The core elements of this concept are presented below. [7]

- **Business Process:** A set of one or more linked activities that collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships.
- **Business Process Modelling Notation:** A standard set of graphical shapes and conventions with associated meanings that can be used in modelling a business process.
- **Process Definition:** A representation of a business process in a form that supports automated manipulation, such as modelling or enactment by a process management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data.
- **Process Model:** A simplified or complete process definition created to study the proposed process before execution time.
- **Process Diagram:** A visual explanation of a process definition.
- **Process Instance:** A data structure that represents a particular instance of executing a process. It has associated context information that can be used and manipulated by the process. A process instance plays a role in a BPMS that is very similar to but not exactly the same as a case in a case management system. A particular case may have more than one process instance associated with it.

- **Activity:** A description of a piece of work that forms one logical step within a process. It is the basic unit of work within a process. Presumably, work could be subdivided into units smaller than a given activity, but it is not meaningful for the organization to track work on that level of detail.

In the context of developing IT systems, business process orientation is well suited for modelling and executing predictable and repeatable sequences of activities. It is typically combined with Service-Oriented Architecture (SOA), which enables the management of a portfolio of web services. [8] These web services are then referenced by activities in the executable process model.

2.5 Knowledge-intensive Processes

“Engineering of knowledge-intensive processes is far from being mastered, since they are genuinely knowledge- and data-centric, and require substantial flexibility, at both design- and run-time.” [9]

Knowledge-intensive processes are positioned in the largely unexplored intersection between the fields of BPM and knowledge management (KM). [9] Unlike business processes, which depict predictable and repeatable routine work, knowledge-intensive processes often include innovative and creative activities instead of structured working practices, depending heavily on the application of human knowledge. [10] Their tasks, event flow and sequence of activities are therefore not clear from the beginning, cannot be precisely defined and can evolve as the process progresses. [9]

According to Davenport, a classification of knowledge-intensive processes can be made on the factors of complexity and interdependence. [11] As shown in the figure below, work is either routine or based on interpretation or judgement, and can be performed either by individual actors or collaborative groups.

A classification structure for knowledge-intensive processes

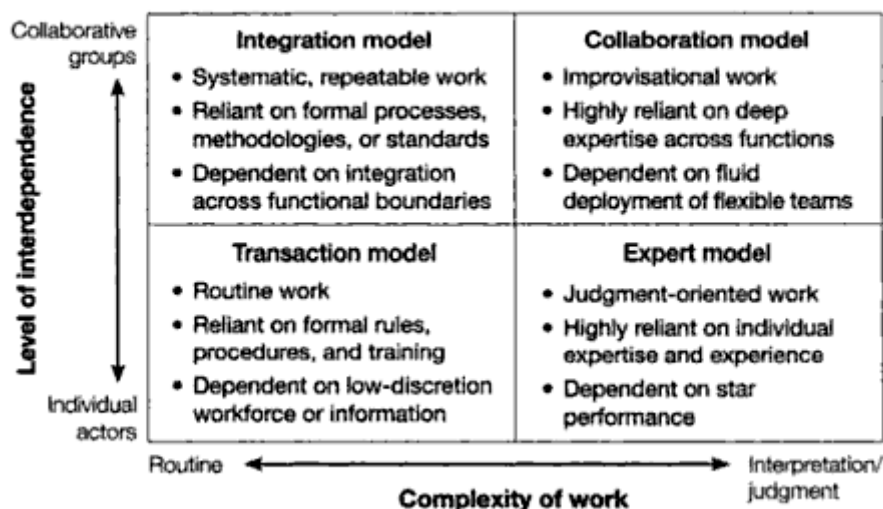


Figure 2 Classification Structure for Knowledge-intensive Processes [11]

The transaction model addresses the simplest scenario, and often features traditional application programming enhanced with rule-based engines or process execution. The integration model represents the classic BPM approach for distributed workflow management, while the expert model relies on browsers, document repositories and office automation tools. Finally the collaboration model serves processes with both high complexity and high interdependence, and usually relies on email as the central technology. [12]

Contrary to Davenport's classification, Ukelson offers his own classification based on a single axis – the process structure – and containing the following classes. [13]

- **Structured process:** A rigorously defined process with an end-to-end model, which takes into account all the process instance permutations. No process instance can stray from process model, similar to structured data, where there is a specific data model associated with the data, and the data cannot stray from that model, and if it does stray, the data is invalid. Structured process can be automated, which greatly shortens processing cycles by eliminating the need for human intervention in the execution of a process.
- **Semi-structured process:** A process in which a portion of the process is structured, and sometimes unstructured processes are invoked (during exceptions, or when the model doesn't hold).
- **Unstructured process:** Every instance of the process can be different from another based on the environment, the content and the skills of the people involved. These processes may have a framework or guideline driving the process, but only as a recommendation.

Several process automation techniques have been introduced in recent years to meet the needs of knowledge workers dealing with unstructured processes. Human processes depend heavily on the interactions between people, and are by nature unstructured and dynamic. [14] Expert Processes are goal-oriented, collaborative and deadline-driven, and lack a pre-defined sequence of activities. [15] Agile processes give business analysts a faster way to design flowcharts, based on more powerful design tools and business process management system capabilities. Dynamic processes enable users to make changes to the process during execution, e.g. by selecting subprocess definitions, but those changes are not propagated to the process definition itself and are thus lost after execution. Ad-hoc processes are simplified processes which only contain a limited number of steps, and focus on a quick fix over long-term sustainability of the approach. Finally, adaptive processes are centred on internal changes caused by outside conditions, which become permanent as the process evolves to meet the outside conditions. This enables knowledge workers to collaborate and create detailed and complex processes on the fly, drawing on available data structures and business rules to describe well-defined goals. [16]

“Adaptive BPM refers to internal changes caused by outside conditions, that become permanent and make the entity more fitting to those new conditions. Those changes are performed by means of the entity itself and not by some external force. With Adaptive Processes, end-users do not just collaborate in flowchart design, but they actually create the real-world process on the fly. Not just a simple ad-hoc activity, but with substantial complexity using metadata models from the repository and business rules in natural language for well-defined goals. Being adaptive is not about predicting how a process will work or to agree on all possible mutations. Adaptive means that real-time knowledge from the last process execution can influence the execution of the next.” [16]

2.6 Domains of Predictability

“Being adaptive is not about predicting how a process will work or to agree on all possible mutations. Adaptive means that real-time knowledge from the last process execution can influence execution of the next.” [16]

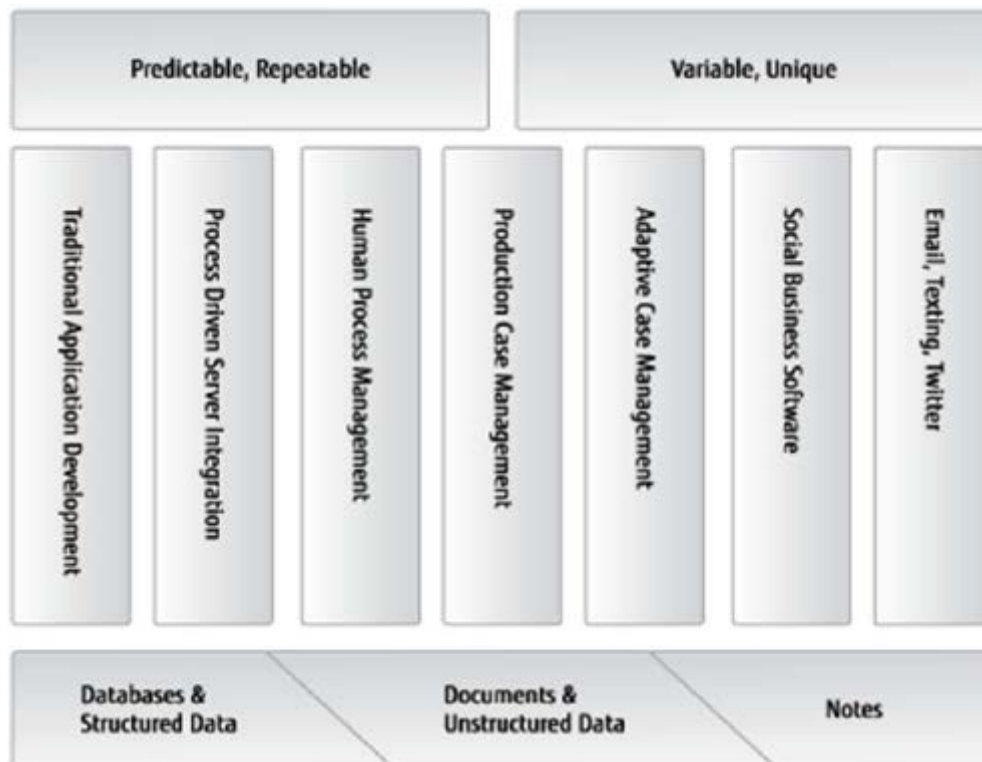


Figure 3 Domains of Predictability [17]

Swenson proposes a classification of process management approaches based on predictability, and goes on to state that predictability and repeatability go hand-in-hand: if work is frequently repeated, it is by definition predictable. The core question when developing a case management system is how much change the system will have to respond to over time. Stable environments are well-suited for inflexible approaches that emphasize automation. But a frequently changing environment benefits from greater flexibility and the ability to adapt to the changed environment. Swenson defines seven domains of predictability as follows. [17]

- **Traditional Application Programming:** If work is very predictable and stable over time, one can use traditional development techniques to create a supporting application. The cost of development may be high, but the benefit of having very precise control [...] will yield efficiency that over a large number of cases will pay back the up-front costs.
- **Process-Driven Server Integration:** Integration patterns between systems can be quite stable in the short term. Still the systems being integrated do change, and the business needs change as well. PDSI usually incorporates a development approach where the key interaction points are depicted on a process diagram.

- **Human Process Management:** Most human processes are executed using standard office technology, yet they are not managed by that technology, but rather through standard management techniques with no, or minimal, system support. This style of integration usually involves forms routed through a set of people. The major shape of the process and the tasks may be well-understood, but HPM has capabilities to handle the kinds of continual change that exist in an organization.
- **Production Case Management:** This is designed to handle situations where there is so much variation between individual cases that it is not possible to set out a single fixed process, and yet there is still a well-known set of actions that can be taken. The knowledge worker is actively involved in deciding the course of events for a case, but the range of actions and options is bounded and can be specified in advance.
- **Adaptive Case Management:** These cases vary so much that knowledge workers are constantly striving for innovative approaches to meet the needs of new cases. The knowledge worker is involved not only in the case, and picking a predefined action, but is actually helping invent the actions that can be taken. However, there is still enough predictability that a given knowledge worker may want to re-use a process from before, and may want to share and discuss process plans with others.
- **Social Business Software:** This is collaborative software, and it includes basic document management systems without a fixed plan. There may be representations of goals, but they are created on the fly and discarded after use.
- **Email, Texting, Twitter:** There is no process at all, no permanent structures, only communication. This is the default that many current processes are forced to use, but this approach puts the greatest burden on the user, and yields the least amount of analytic data to monitor and improve processes.

The work performed by knowledge workers across various business domains at the EC falls somewhere between the extremes of traditional application development and email/text/twitter. Identifying the level of predictability in a given business domain, and choosing the appropriate approach, is the first step to developing a case management system, and is further discussed in chapter 4.2.

2.7 Case Management Approaches

“A method or practice of coordinating work by organizing all of the relevant information into one place – called a case. The case becomes the focal point for assessing the situation, initiating activities and processes, as well as keeping a historic record of what has transpired.” [18]

While the term case management is often used to refer to applications in specific business domains like legal, healthcare or social services, it can in fact be applied to any domain involving knowledge work. Case management removes the focus on pre-defined sequences of activities found in business process management, allowing the information collected during a workflow to become the main driver for decision making. This allows case management to address domains of low predictability, and offers new approaches to supporting knowledge workers with IT systems. The key terminology of case management is defined as follows. [5]

- **Case:** The name given to the specific situation, set of circumstances or initiative that requires a set of actions to achieve an acceptable outcome or objective. Each case has a subject that is the focus of the actions – such as a person, a lawsuit or an insurance claim – and is driven by the evolving circumstances of the subject.

- **Case Owner:** A person (or group of people) who is responsible for the outcome of a case. The case owner can change any aspect of a case and is actively involved in achieving the goals of the case.
- **Case File:** Contains all of the case information and processes, and coordinates communications necessary to accomplish the goal for a particular case. A case file can contain information of any type including documents, images, video etc.
- **Case Plan:** A written document that identifies the goal of the ongoing case and the outcomes and actions required to achieve the goal.

This thesis will focus on three approaches to case management: business process management, as the approach followed by both the NEST prototype and the Customs Decisions (CD) project; production case management, as an approach for semi-predictable processes relying on a well-known set of actions; and adaptive case management, as an approach for unpredictable processes relying on the knowledge worker himself to determine the actions to be taken.

2.7.1 Business Process Management

The practice of developing, running, performance measuring and simulating business processes to effect the continued improvement of those processes. [18]

The focus of BPM is the business process, and it uses the process as an organizing paradigm around which data, roles and communications are organized. Process models are prepared in advance for particular situations, and the performance can be measured and monitored so that over time the process will be improved.

2.7.2 Production Case Management

An approach to supporting knowledge workers which is programmed by specially-trained technical people, and offers collections of operations that a knowledge worker may choose to use, depending on the specific needs of the case. [17]

A PCM system is used when there is a certain amount of unpredictability in the work, and some flexibility is needed, but necessary actions are regular enough or the volume of work is large enough to make identifying and codifying regular patterns valuable.

2.7.3 Adaptive Case Management

A productive system that deploys not only the organization and process structure, but becomes the system of record for the business data entities and content involved. [7]

In his books on supporting unpredictability in knowledge work, Swenson observes that: the focus of ACM is the case information, and all other artefacts are organized around it; it is the case information that persists for the long term; all processes are completely transparent, as per access authorization, and fully auditable. [7] Pucher notes that: ACM enables non-technical business users in virtual organizations to seamlessly create/consolidate structured and unstructured processes from basic predefined business entities, content, social interactions and business rules; ACM moves the knowledge-gathering process from the template analysis phase to the process execution phase in the lifecycle; ACM collects actionable knowledge – without an intermediate analysis phase – based on process patterns created by business users. [19] Finally Kraft identifies the characteristics of an ACM system as follows. [20]

- **Goal-oriented:** The workflow aims to achieve a goal.
- **Unpredictable:** The workflow cannot be defined a priori.
- **Guided:** Templates guide the emerging workflow.
- **Constrained:** Policies constrain the emerging workflow.
- **Shared:** Best practices are shared within the community.
- **Tracked:** The remaining effort necessary to fulfil the goal is tracked.
- **Visible:** The performed work is visible to managers and co-workers.
- **Automated:** Automating predictable parts of a workflow frees up time for creative tasks.

2.7.4 Comparison of Case Management Approaches

There is no one-size-fits-all solution, and indeed each of the three approaches described above works best for some specific scenario. As discussed previously, predictability is one of the defining criteria in choosing a case management approach. Other criteria include case volume, domain knowledge and worker skill. A comparative analysis of the three case management approaches presented above, as well as a discussion on appropriate modelling notations and process execution engines follows below.

	BPM	PCM	ACM
Predictability	High	Medium	Low
Case volume	High	High	Low
Domain knowledge	High	High	Low
Worker skill	Low	Medium	High
Modelling notation	BPMN	BPMN	BPMN/CMMN
Execution engine	Activiti	Activiti	Customised Activiti

Table 1 Comparison of case management approaches

2.7.4.1 Predictability

Predictability is such an important criterion, because of the significance it has for process modelling and execution. BPM originated in the field of factory automation, where input (raw resources), output (end product) and process (manufacturing technique) leave little room for unpredictable events. If something does go wrong during process execution, the process (and thus the machine) halts and waits for human intervention to find and fix the error, before either continuing where it left off, or starting again from the beginning. The very same pattern holds true for BPM-based distributed workflow management engines like Oracle BPM. If something unforeseen happens, manual human intervention is the only way to fix a broken process instance.

Similar to BPM, PCM assumes that everything that can happen is known at design-time. It offers flexibility through choice, but falls short of the ability to adapt to unforeseen circumstances. ACM goes a step further, as it allows knowledge workers to handle situations that have not been anticipated in advance. This flexibility comes at a price though, as the amount of control (through definition of all possible actions at design-time) present in BPM- and PCM-based IT systems is sacrificed.

2.7.4.2 Case Volume

Case volume deals with the sheer number of cases of a certain type, which an organisation has to handle. The higher the number of instances of a certain case type, the more feasible it is to analyse, model and implement a business process to describe that case type. Both BPM and PCM rely on high case volume, to justify the development cost of a process model. ACM on the other hand shines in domains with low case volume, as the actual creation of a case model is deferred to the execution phase, and carried out by the knowledge worker himself.

2.7.4.3 Domain Knowledge

Domain knowledge refers to the amount of knowledge an organisation has about a business domain. Both BPM and PCM require a high level of domain knowledge, as all possible future events have to be anticipated at design-time. One can in fact make all the right decisions when designing and implementing the IT application part of a case management system, only to see a lack of quality of the process models cause the project to fail. Business processes are an effective approach to orchestrating IT services, but in the end, business process-oriented IT systems depend on business analysis and process modelling teams to provide the actual content.

ACM takes a different approach by moving the case modelling from the design phase to the execution phase. This puts the onus of modelling on the knowledge worker handling the case, and makes the classic business analysis and process modelling teams obsolete. On the one hand, having the knowledge worker perform the business analysis (i.e. identifying what can be done and what should be done) may very well be an improvement, as – unlike most business analysts – they have first-hand knowledge of the business domain they are working in. On the other hand, relying on knowledge workers to model a case at run-time, may require a level of IT skills that an experienced trade negotiator, competition lawyer or customs officer simply lacks.

2.7.4.4 Worker Skill

The worker skill criterion follows the opposite reasoning of predictability. The more predictable a case is, the more control can be applied at design-time. The application of more control at design-time leads to a lower level of control at run-time, which in turn requires less skill on the part of the knowledge worker handling a case. BPM restricts a knowledge worker to making decisions that have been defined at design-time, while PCM offers a knowledge worker the flexibility to choose from a set of possible solutions at run-time. ACM on the other hand allows a knowledge worker to identify the correct solution from his own experience, or come up with a new solution at run-time. This requires a high level of skill on the part of the knowledge worker, and poses a challenge to ACM systems which cannot rely on knowledge defined at design-time to guide the knowledge worker.

2.7.4.5 Modelling Notation

Two modelling standards are specifically relevant to case management. The Business Process Model and Notation (BPMN) allows for the modelling of fully predictable and structured business processes, while the Case Management Model and Notation (CMMN) addresses the need to model semi-predictable and -structured cases. Both modelling notations are presented in chapter 2.8, which also includes a discussion on why neither fully satisfies the requirements of ACM. The key element missing is the possibility of choosing a solution which is not known at design time. An interesting approach to overcome the limitations of design-time modelling is outlined by Pucher, who suggests building a

collaborative system around the process execution engine, which manages case templates and goal definitions, and enforces certain temporal and logical rules on the activities defined at run-time. [21]

2.7.4.6 Execution Engine

While many vendor-based solutions for business process execution exist, proprietary software prevents the adaptation of existing solutions. One of the leading open source solutions for process execution is Activiti, which is based on the BPMN 2.0 standard for business process modelling. The out-of-the-box implementation of Activiti allows for execution of both BPM and PCM processes. As discussed in chapter 6.6 however, a minor adaptation of the Activiti implementation could allow for the level of flexibility required by ACM.

2.8 Process and Case Modelling

The Object Management Group (OMG) has specified two standards for modelling business processes and cases. BPMN, first released in May 2004, defines an explicit logical sequence in which activities occur in a process. CMMN, first released in May 2014, relies on events and conditions to steer the flow of a case. While BPMN offers only limited support for ad-hoc activities in process models, CMMN allows for dependencies based on temporal logic. [6]

Hinkelmann and Pierfranceschi have shown that neither BPMN nor CMMN alone are adequate for the kind of adaptive processes that public administrations deal with. [22] While BPMN lacks support for run-time planning, CMMN fails to provide a visualisation of execution flow and responsibilities. They propose a combination of the control flow elements of BPMN with the adaptive planning elements of CMMN, as a solution for visually representing adaptive processes.

2.8.1 Business Process Model and Notation

The Business Process Model and Notation (BPMN) specifies a metamodel, notation and interchange format for modelling collaboration, process and choreography diagrams. It aims to be readily understandable by all business users (e.g. analysts, developers, system owners), and attempts to bridge the gap between business process design and implementation. While collaboration and choreography models describe the interaction between processes, the process model defines a set of flow elements, as illustrated in the figure below.

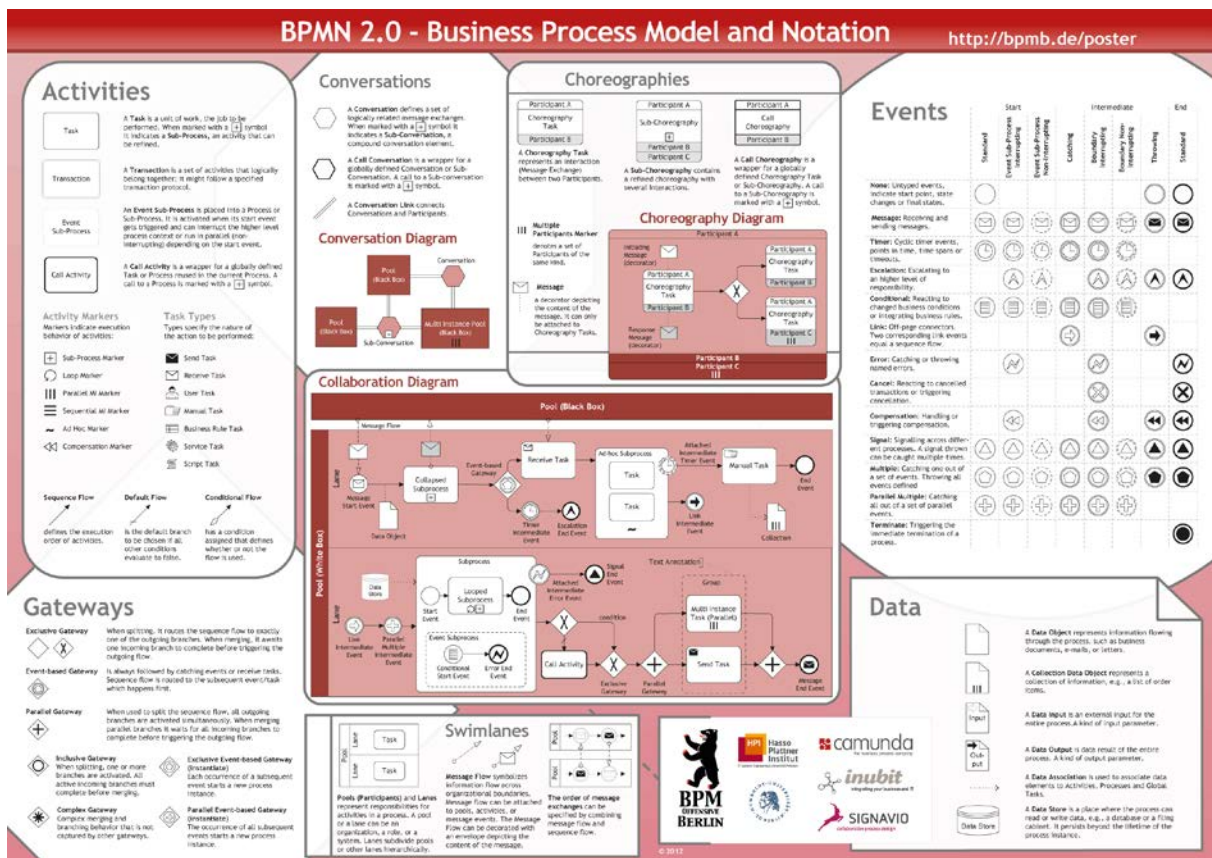


Figure 4 BPMN 2.0 - Business Process Model and Notation [23]

The key element of BPMN is a process, which acts as a container for various elements. A process has one or more defined start and end points, and contains a sequential flow of elements leading from one or more start points to one or more end points. A process model is defined during design-time, and executed during run-time. Changes to a process model at run-time are not allowed, though some business process execution engines allow migration of the process model to a newer model version during run-time. BPMN defines the following list of elements. [24]

- **Process:** A process is a top-level concept that combines all elements constituting a process model.
- **Activity:** Activities include tasks, transactions, subprocesses or call activities. An activity constitutes a unit of work performed within a business process. A task can be a manual human task, or the automated sending or receiving of a message, or execution of a service or script. A transaction is a set of activities that logically belong together, and whose result is only committed to the process state if the transaction completes successfully. A subprocess is a recursive concept which follows the same rules as a process (i.e. defined start and end points connected by a sequential flow of activities). A call activity is a wrapper for a globally defined task or process, and enables reusability of model fragments.
- **Flow:** Flows define the execution order of activities. A sequence flow defines a branch that is always taken. A conditional flow defines a branch that is only taken if a

specified condition is met. A default flow is a branch that is chosen if the conditions of all other potential branches are not met.

- **Gateway:** Gateways control the flow of a process by splitting or merging it. A parallel gateway splits the flow and activates all outgoing branches, or merges the flow while waiting for all incoming branches to complete. An exclusive gateway splits the flow and activates exactly one outgoing branch to follow, or merges the flow while waiting for exactly one incoming branch to complete. An inclusive gateway splits the flow and activates one or more outgoing branches, or merges the flow while waiting for all active incoming branches to complete. Event-based gateways are based on catching events rather than evaluating conditions, and complex gateways are a catch all for any splitting and merging logic that cannot be expressed through the other gateway types.
- **Event:** Events are occurrences that affect the flow of a process. An event can be caused by or trigger various circumstances: creation of a process instance (i.e. start event); completion of a process instance (i.e. end event); sending or receiving of a message; elapsing of time; escalation of responsibility; change in the process state; cancelling of transactions or subprocesses; handling or triggering of compensation; catching or throwing of errors; and signalling across process instances.
- **Artefact:** Artefacts provide the capability to express information associated with an activity. A data object represents information flowing through a process, such as documents or e-mails. Data input and data output specify input and output parameters of a process or task. A data store defines a storage space (e.g. database) for process data, which persists beyond the lifetime of a process instance.
- **Swimlane:** Swimlanes are a recursive concept enabling the hierarchical depiction of responsibilities for activities in a process. Each swimlane refers to a systems, role or organisational unit, and contains those activities, which that system, role or organisation unit is responsible for.

2.8.2 Case Management Model and Notation

The Case Management Model and Notation (CMMN) specifies a common meta-model and notation for modelling and graphically expressing a Case, as well as an interchange format for exchanging Case models among different tools. Following the success of BPMN, which over the past 13 years has become the de facto standard for business process modelling, CMMN aims to satisfy the need for modelling unpredictable and non-repeatable activities, which depend on evolving circumstances and ad-hoc decisions.

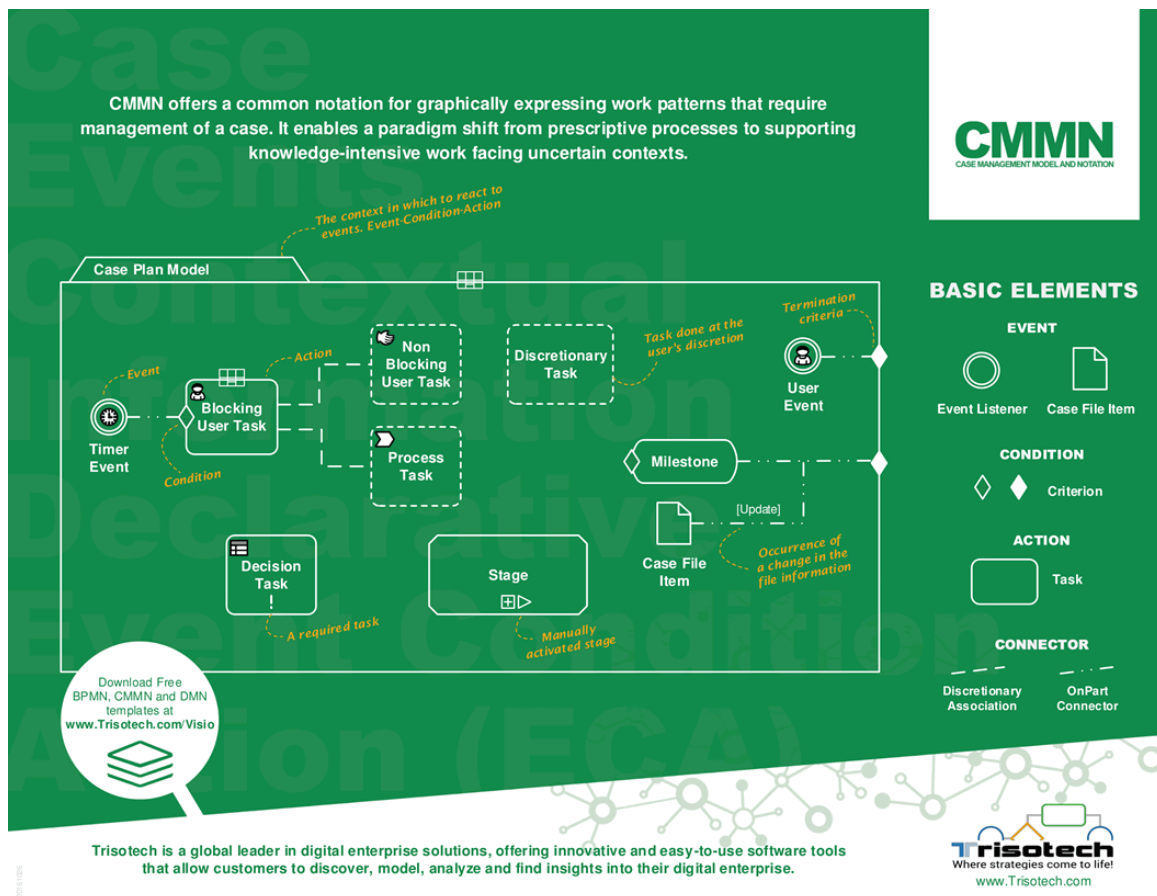


Figure 5 Case Management Model and Notation [25]

The key element of CMMN is a case, which acts as a container for various elements. A case is split into two phases: during design-time, a case plan is created by business analysts, who try to capture the expected flow of activities for a certain case type; during run-time, the case plan is adapted by choosing to ignore existing elements or adding new elements based on the emergent needs of a case. CMMN defines the following list of elements. [26]

- **Case:** A case is a top-level concept that combines all elements constituting a case model.
- **File:** A file collects all structured or unstructured information (and references to such information) required in the context of managing a case. It represents the case state, and holds all data and documents related to a case, as well as optional metadata describing those contents. It serves as the single source of information for events, expressions and parameters.
- **Plan:** A plan is a container for all case activities planned during both design-phase and at run-time, including events, milestones, tasks and stages. Similar to the root process in BPMN, every case has a single outer case stage containing all plan items.
- **Stage:** A stage is a decomposition of a case plan, which can contain plan items and sentries. It is a recursive concept that can be further split into sub-stages.
- **Task:** A task is an atomic unit of work, performed either manually by a human, through an automated decision, or (similar to a call activity in BPMN) by another case or business process.

- **Planning table:** A planning table is an optional decorator on stages or tasks, which indicates: a set of plan items (i.e. discretionary items) that can be considered during run-time planning of that stage or task; a set of rules that regulate the applicability of those plan items; and a set of roles defining the users that are authorised to perform the planning of that stage or task during run-time.
- **Milestone:** A milestone represents an achievable target, defined to enable the evaluation of case progress. Milestones are usually related to the completion of tasks or the capturing or creation of information in the case file.
- **Event:** An event may trigger actions like enabling, activation or termination of stages or tasks, or the achievement of milestones. Case events have various causes, including standard events (e.g. changes to the case state, stages, tasks or milestones), timer events (i.e. a pre-defined amount of time elapses), and human events (i.e. manually triggered by a business user).
- **Sentry:** A sentry listens for the occurrence of important events which influence the further proceedings of a case, and represents the events and conditions required for enabling tasks, stages or milestones, and completing or terminating cases, stages or tasks. Sentries combine the concepts of event and/or condition, firing when one or more specified events occur, and/or one or more specified conditions apply.
- **Role:** A role authorises a user or group of users to trigger events, complete tasks, or adapt the case plan by planning discretionary tasks.
- **Parameter:** A parameter models the input or output of a case or task. It can be used to exchange information both within a case (e.g. initiating a task to draft a document with a document template), and to or from the outside (e.g. providing a document containing a legal decision as the result of a completed case).
- **Expression:** An expression can be used to automate decisions based on evaluating elements of the case state, as well as constants or time.

3 Core Requirements

Having established the theory and terminology behind Case Management, this chapter focuses on the Adaptive Case Management (ACM) approach and discusses the core features of an ACM system. We will start with Max J. Pucher's 2010 definition of "The Elements of Adaptive Case Management" [19]. Next we will look at "Leveraging CMMN for ACM: examining the applicability of a new OMG standard for adaptive case management" [6], a 2015 publication examining how ACM requirements can be mapped to the Case Management Model and Notation (CMMN), a recently released standard for case modelling. We will then examine the ongoing Case Management Rationalisation (CMR) exercise at the European Commission, which aims to provide a new and efficient case management system for several Directorate-Generals. Finally, the requirements identified by Pucher, Kurz et al. and CMR are compared, and a set of core requirements for the ACM prototype is selected.

3.1 Elements of Adaptive Case Management

In his contribution to Keith Swenson's 2010 book "Mastering the Unpredictable" [19], Max J. Pucher, one of the leading proponents of Adaptive Case Management (ACM) and co-founder of ISIS Papyrus, discusses the core elements of ACM. As "The Forrester Wave™: Dynamic Case Management, Q1 2014" noted, ISIS Papyrus is a "standout leader in run-time tool weighting" [27]. Its framework focuses on flexible run-time behaviour, and an emphasis on business terminology and goal orientation serve to empower knowledge workers. This

approach is representative of Pucher's rejection of traditional Business Process Management (BPM) as a panacea for increasing productivity in service-oriented businesses.

“How many processes of a business can be optimized is a matter of perspective and opinion: between 20% and 80% [...] – I lean more toward 20%.” [19]

Pucher is adamant that the differences between highly predictable manufacturing work and highly unpredictable knowledge work mandate a different approach. Where the infinite repeatability of a manufacturing process perfectly suits the idea of an infallible workflow to be executed over and over again, knowledge work will suffer from too much automation, which by definition reduces the knowledge worker's ability to apply his skills to unforeseen situations.

“The needs of these knowledge workers require a technological empowerment rather than a new management methodology. While the business strategy is a top-down definition targeting cost, process innovation targeting quality is most likely more effective bottom-up. Technology has to support both targets – cost and quality – to be acceptable and beneficial to management employees and customers.” [19]

3.1.1 Lifecycle

By moving the knowledge gathering process from the analysis phase into the execution phase (labelled “Process monitoring and controlling” in the figure below) of the process lifecycle, Pucher suggests that business users can effect incremental, low-cost process improvements, whose benefits can be verified within days, instead of months or years. [19] The goal is an IT system that allows non-technical business users to seamlessly create and consolidate both structured and unstructured processes from pre-defined business entities and rules, content and user interface components.

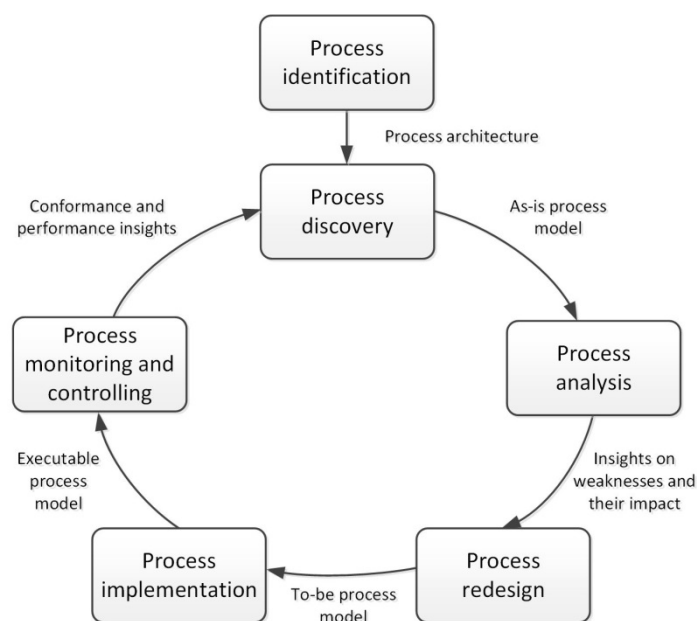


Figure 6 Business Process Management Lifecycle [28]

Unlike traditional BPM, there is no need for an elaborate project management infrastructure to handle the analysis, modelling, simulation, and implementation of a new business process.

While the focus on lower costs and higher system stability has led to an inability to adapt and innovate, and six-month rollout cycles following the waterfall methodology are still very much the norm in public administration IT departments, ACM promises to shorten change management cycles, and empower business users by giving them the means to satisfy their customers. Instead of merely allowing execution of a workflow designed by an army of business analysts as result of a multi-year process landscaping effort, ACM enables business users to add knowledge to a process model during run-time, thus adapting the planned workflow to the circumstances of an unforeseen situation. Rather than providing a specific solution to a specific problem, ACM provides a toolset to solve a certain class of problems. To empower business users to act as knowledge workers, Pucher suggests that ACM has to provide the following capabilities. [19]

- Ability to manage and adapt processes
- Mapping between master data models, business rules, business processes and service interfaces
- Classification, extraction, validation and publication of content
- Authenticated access control on both entity and functional level
- Auditability of templates and definitions
- Transparency through secured access to critical information

3.1.2 Case Templates

Pucher considers the creation, deployment, and maintenance of case templates (i.e. process models) to be a core issue of ACM. [19] A central model repository should provide collaborative and transparent change management capabilities, enabling evolution, variation and adaption of process models. The repository contains the actual process models, as well as metadata captured to document the models and make them retrievable via search. Based on this model repository, an ACM system provides the required functionalities for collaboration and transparency. As with traditional BPM, the re-use of existing models can guide the case team and eliminate the need to reinvent the wheel. However, the ability to change the process model at execution-time provides an inherent advantage to ACM.

Consider a traditional Business Process Management System (BPMS). Once the system is operational, any change to the workflow requires a change request to the systems provider, followed by a lengthy discussion and analysis phase to determine the feasibility of the change request. This in turn is followed by a contractual phase, development, testing, etc... Providing business users with the ability to perform changes to the workflow themselves enables the kind of flexibility that would be suffocated by a traditional change management process. There are two limits to this flexibility. Firstly, similar to a toolbox, the service landscape defines the possible actions that can be selected in a process model. If a new action would require a service that is not available in the organisation's service catalogue, the business user once again has to rely on the traditional change management cycle to provide it. Secondly, business users require training to work with process models, a factor which is often underestimated, as system providers' assurance that adequate training will be provided, are juxtaposed by the reality that business users in public administration tend to have more legal expertise than experience in operating complex IT systems.

3.1.3 Decision Making

“Because of our human ability to memorize data in sequential chains that we infer to be causal, we fail to see the complexity that appears when we fragment information into data, processes and rules.” [19]

Pucher argues that data experts fail to see the importance of processes in the interpretation of data, the same way that process experts overvalue the significance of pre-defined sequences of actions. [19] ACM has to provide a big-picture approach, which brings together various knowledge silos to provide a unified, meaningful view. A Business Rules Engine (BRE) is often used as the decision-making component of an IT system, generating complex results from large sets of data and rules. Unlike the traditional BRE approach of separating business logic from process logic however, Pucher advocates the importance of context (i.e. the process state) to rules. [19] The state of a process may have a significant influence on the resolution of the business rule itself, necessitating that rules have to be evaluated in the context of a specific process instance. Furthermore, the ability to change rules in a local context allows business users to effect local changes without risking adversary effects on other processes that may rely on the same business rules. Finally, it is essential to synchronise processes and rules, as changing either may have a direct impact on the other.

Pucher goes on to point out that process goals are in fact rules, not attributes. [19] Let us assume a very simple case, whose goal is to provide a case report, i.e. a document that outlines the findings of an investigative case. In BPM, a pre-defined workflow would define a series of actions leading to the inevitable goal, the creation of a case report. In this case the goal is an attribute of the process that will be assigned once the workflow is completed. In ACM however, the case can be seen as an ongoing dynamic process which, regardless of the effective sequence of actions, ends after the goal is reached, i.e. after a case report is created. In this case the goal is a rule stating that a case report must be provided. When this rule evaluates to true, the goal is reached and the process is completed.

3.1.4 Content

Pucher states that a process is meaningless without content, and content is useless without a process. [19] Content is certainly a central element of case management in a public administration, where written documents and forms serve both as input and output of a case. Put in the terms of a traditional manufacturing process, inbound content serves as the raw resources, which the case process turns into a finished product, in the form of outbound content. Inbound content can arrive in various formats, e.g. electronic and paper documents, forms, emails, archives, semi-structured data, binary files etc... The process transforms these representations based on process models and business rules, and generates new content made available in similar formats.

3.1.4.1 Inbound Content

The automated capture of inbound content is one of the primary challenges of ACM. [19] Given enough structure (e.g. a form bound to a well-structured entity, or an XML schema defining a semi-structured entity) capturing content is fairly straightforward. But unstructured content like documents and emails may at best contain only a small set of metadata. And paper content requires scanning, which creates manual work, and automated text recognition, which can have a severe negative impact on the quality of the captured data. [29] Furthermore, large volumes of inbound content require automated classification to avoid creating a bottleneck before the actual knowledge work even starts.

3.1.4.2 Outbound Content

Standard document management solutions provide lifecycle functionality, which can be transparently integrated into an ACM system. The major challenge of outbound content in ACM thus lies in the merging of the process state with a document template. [19]

- Data sources have to be exposed via service interfaces.
- The creation of a document has to be initiated by a process or user.
- Document templates have to be provided, managed and verified, before they can be merged with the process state to create a document.
- Created documents have to be reviewed and approved, before they can be published.
- Electronic publication requires access control, digital signature and an audit trail that tracks events like creation, modification and approval.
- The visual identity of an organisation has to be enforced through custom headers, footers and watermarks.
- Last but not least, information security policies have to be considered if the content produced is confidential.

3.1.4.3 Record Management

ISO 15489-1:2016 defines the term record as *“information created, received and maintained as evidence and as an asset by an organization or person, in pursuit of legal obligations or in the transaction of business”*. [30] In any large organisation, but especially so in public administration, the information sent or received has to be stored as evidence. Such records usually have legally-binding retention policies, defining how long an organisation has to be able to provide them in case of an audit or a legal investigation, and an ACM system must cover the management of these retention policies. Digital signature is required to prove authenticity, and records must be stored safely, as loss of data is not an option, and securely, to prevent unauthorised access. Finally, an ACM system has to integrate existing record management systems, as an organisation's policies may very well require a specific system to be used for certain records.

3.1.5 Security

ACM offers an integrated solution in an organisation-wide user environment, resulting in the need for authenticated access control on both entity and functional level. [19] All users have to be authenticated (i.e. their identity is confirmed). All entities (e.g. processes, content or data) have to be whitelisted for specific users or groups. All functionalities (e.g. viewing, modifying or deleting) have to be restricted to roles, which are assigned to specific users or groups. User and group management requires not just the initial creation, but also curation over time, as users change or leave groups (thus gaining or losing access rights), and new groups are formed or existing groups closed (e.g. when organisational units are reorganised or temporary interest groups expire).

3.1.5.1 Authentication

Any acting user of an ACM system must be an authenticated actor (i.e. an actor whose identity has been verified), as even the most basic act of viewing information has to be secured. [19] The simplest form of verification is a password, a logical key that only the actual person associated with the actor's username would know. The success of this method depends on several factors.

- The password must be created either by the system or the actor himself, but not by another person.
- Electronic transmission and storage of the password must be secured by encryption.
- The actor must not share the password with another person, or store a physical copy (e.g. written down on a piece of paper) in a location that another person has access to.

Another method of authentication is a combination of logical and physical key, e.g. a smart card. All information on the card (i.e. physical key) is stored in encrypted form, and the actor has a decryption key (i.e. logical key) to access the information stored on the smart card. The system reads the information on the smart card using the decryption key provided by the actor, thus ensuring that he provides both a logical and a physical key to identify himself.

A third method of verification is biometric. This can be combined with a logical or physical key to add additional security layers, but the point is always to confirm the physical presence of a specific person. It is important to note that no method is perfect: logical keys can be broken; physical keys can be stolen; and biometric keys can be faked. Rather than trying to find a single perfect authentication method, a combination of multiple authentication methods is usually applied to increase the security level.

3.1.5.2 Authorisation

Authorisation defines the actions an identified actor can perform in an IT system, and ACM requires authorisation on both functional and entity level. [19] Functional authorisation is usually achieved through Role-based Access Control (RBAC). RBAC associates actions (i.e. a service method) with roles, which in turn are associated to users and/or groups of users. This ensures that only an actor with a required role may perform a certain action, i.e. that an actor with role <A> may perform action . A popular method of authorising access to entities is resource-based access control using an Access Control List (ACL). ACLs are lists of 3-tuples, associating a user or group of users with an object (e.g. data entity) and an operation (i.e. action). This ensures that only a certain set of actors may perform a certain set of actions on a specific object, i.e. that actor <C> may perform action on object <D>.

3.1.5.3 Encryption

All sensitive information should be transmitted and stored in encrypted form. [19] HTTPS is slowly becoming a standard to ensure the privacy and integrity of transmitted data, using Transport Layer Security (TLS) or Secure Sockets Layer (SSL) to encrypt communication between client and server. Database encryption deals with securing the sensitive information entered into a system. As databases are separate components, and the system itself can only control access to its own user interface, both authorised (e.g. by developers or database administrators) and unauthorised (e.g. a malicious hack or accidental leak of the database content) access to the information stored in the database (through another method than the secured user and service interface provided by the system) has to be prevented.

3.1.5.4 Confidentiality

A widely accepted method for the management and distribution of encryption and decryption keys is public key cryptography using a Public Key Infrastructure (PKI). This method relies heavily on the non-existence of efficient solutions to certain mathematical problems (e.g. integer factorisation, discrete logarithm, and elliptic curve relationships), and it would be cause for great concern to the IT security community, if it were ever proven that, if the

correctness of a solution to a problem were easy to check, the problem would be easy to solve.

A pair of keys – a public key known to everyone, and a private key known only to the actor – is used to ensure both confidentiality (i.e. the content can only be decrypted by the actor) and authenticity (i.e. the identity of the actor). Confidentiality is based on the actor using his private key to decrypt a message, which can be encrypted by anyone using the actor's public key. Authenticity is based on the actor using his private key to encrypt a signature, which can be verified by anyone using the actor's public key.

3.1.5.5 Authenticity

While the legal validity of electronic signatures is established, no technology standard has been defined. A popular solution is the above described PKI, which can ensure authenticity using public and private keys, issued by trusted Certification Authorities (CA). The signature of the actor is a central element in any legal context, and the lack of digital signature capabilities requires the scanning of a physically signed document for every legally relevant transaction. It is important to note that legally valid proof of authenticity is only provided by authentication methods that follow the applicable legal regulation, e.g. the Electronic Identification, Authentication and Trust Services (eIDAS) [31] in the EU.

3.1.6 Accountability

Accountability is assured by guaranteeing that every actor is authenticated, and every action can be audited. [19] Authentication provides a reliable identification of the actor, while a mandatory audit trail on all data entities ensures that an audit can identify the actor that performed an action at a specific time on a certain entity. While accountability does not prevent abuse of an IT system, it is vital to security incident investigations and often a legal requirement.

3.1.7 Transparency

A significant advantage of integrated solutions is the range of available information. ACM can achieve transparency by leveraging this range of information to provide secure access to critical information on current and past cases. [19]

- Timeline graphs can document past activities over time.
- Real-time data can help monitor open cases.
- Dashboards provide relevant information to business users.
- Reports keep management up-to-date on the overall case load.
- Statistics can provide valuable information to help improve the quality and efficiency of case workflows.

3.2 Mapping CMMN to ACM

In the 2015 paper “Leveraging CMMN for ACM: examining the applicability of a new OMG standard for adaptive case management” [6] Kurz, Schmidt, Fleischmann and Lederer examine how ACM characteristics and requirements can be mapped to the Case Management Model and Notation (CMMN), a recently established standard for case modelling. The question asked in this article is essentially whether a modelling standard exists for the highly unpredictable processes addressed by ACM. If a modelling standard existed, it should follow that an execution standard can be provided based on the modelling

standard. If we look at BPMN for comparison, version 1.0 was released in May 2004, and there are numerous COTS and open source-based BPMN execution engines available on the market today. CMMN 1.0 was released in May 2014, and so far only Camunda Fox and Confluence claim to support it. If you look under the hood however, you will soon realize that the claimed CMMN coverage is actually quite limited. [32]

3.2.1 ACM Notation Requirements

Based on the ACM characteristics described in “Adaptive Case Management: Supporting Knowledge Intensive Processes with IT Systems” [33], the ACM metamodel introduced in “Taming Diversity: A Distributed ACM-Based Approach for Cross-Enterprise Knowledge Work” [34] and extended in “Subject-Oriented Adaptive Case Management: Extending Subject-Oriented Business Process Management to Knowledge-Intensive Cross-Enterprise Business Processes” [35], the practical review discussed in “Adaptive Case Management in the Social Enterprise” [36], and the theoretical review discussed in “Examining Adaptive Case Management to Support Processes for Enterprise Architecture Management” [37], Kurz et al. propose the following 11 requirements for an ACM notation. [6]

1. **Run-time flexibility:** Based on the ACM characteristic “*flexibility during case execution*” [33] and the ACM requirement “*support ad-hoc changes*” [36], case workers need the ability to adapt all parts of a case during run-time. The unpredictable nature of knowledge work makes the alternative (defining all parts of a case during design time) unfeasible.
2. **Cross-case improvement and case templates:** Based on the ACM characteristic “*continuous improvement*” [33] and the ACM requirement “*provide best practice process definitions*” [36], case workers need the ability to identify best practices in completed cases, and integrate them into the planning of future cases. While run-time flexibility deals with the improvement of a single case through adaption, cross-case improvement addresses the continuing improvement over succeeding cases.
3. **Responsibilities:** Based on the ACM characteristic “*collaboration and transparency*” [33] and the ACM metamodel elements “*proposedTo*” and “*assignedTo*” [34], the responsibility for the various elements of a case must be clear to all case workers. Every action on every process, task, event, document, etc... must be associated with an identifiable user or group of users, both before the action (e.g. when determining who may complete a task) and after the action (e.g. when determining who has completed a task).
4. **Case progress:** Based on the ACM characteristic “*transparency*” [33], all parties involved in a case need the ability to identify which parts of a case are currently active. Case workers need efficient access to the tasks they are expected to perform, and managers need to know when a case they are responsible for is not progressing as planned. Customers usually prefer that their issues are solved yesterday, and seeing how their case is progressing helps them understand “what is taking so long”.
5. **Roles:** Based on the ACM characteristic “*integration of resources*” [33] and the ACM metamodel element “*role*” [34], the ability to adapt access rights during run-time is required. Roles-based access control (RBAC) can provide an indirection layer to facilitate the modification of access rights, but given the existence of viable alternatives like principal-based access control, this concept has to be optional. It is important to note that role- and group-based access control is very similar, in fact the LDAP Data Interchange Format (LDIF) uses the same object class (i.e. `groupOfUniqueNames`) for the definition of both sets of users. This is the weakest of

the 11 requirements presented by Kurz et al., and it could be argued that the key to effective ACM access rights management is resource-based access control, which enables the association of users, groups or roles with actions and entities. This concept is described more closely in chapter 3.1.5.2, and discussed in detail in chapter 7.5.

6. **Objectives:** Based on the ACM characteristic “*goal-oriented*” [33] and the ACM metamodel element “*objective*” [34], the objective of a case has to be clearly expressed, before the progress of a case can be measured. BPM has a rather narrow definition of goal (i.e. to reach an end state), and can use business process simulation to e.g. calculate the shortest path from an active state to an end state. ACM however calls for a flexible goal definition, e.g. to analyse and assess gathered information, or to provide a draft of a legal decision. An ACM system has to provide the means to both define and measure such goals.
7. **Content integration:** Based on the ACM characteristic “*data-centric*” [33] and the ACM metamodel element “*document*” [34], an ACM system needs the ability to integrate tightly with a document and records management system. As case work is document-centric, several case management approaches focus on these artefacts, while providing little or no process execution capabilities. ACM systems should therefore focus on the execution of structured and unstructured processes (for which no standard exists), while integrating with existing standard document management and database solutions to provide the required document and records management capabilities.
8. **Hierarchical task structures:** Based on the key principles of ACM [33], adding new tasks to a task hierarchy is easier for case workers, than embedding new tasks into a complex process. As discussed in detail in chapter 7.2, the hierarchical and recursive structure of processes and tasks allows for powerful enhancements of existing BPM execution technology to satisfy the needs of an ACM system.
9. **Temporal-logic dependencies:** Based on the core concepts of ACM [34], case modellers need the ability to define temporal logic dependencies between the various tasks of an ACM system. While a structured process defines a specific sequence in which tasks are executed, an unstructured process avoids this tight temporal coupling. To influence the order in which tasks in an unstructured process have to be completed, concepts like before, after or in-between have to be supported by the case model.
10. **Interoperability:** Based on the ACM requirement “*exchange events and information with other systems*” [36], even though knowledge work typically has a low level of automation, an ACM system requires the ability to integrate with other existing systems. Service-oriented Architecture (SOA) provides an excellent approach for supplying services, which can be orchestrated and consumed by structured processes. As discussed above, existing standards like document management should be re-used, rather than reinventing the wheel. And certain horizontal domains like authentication, translation, reference data or records management are often provided organisation-wide, making integration of existing systems mandatory.
11. **Legibility and usability:** Based on the ACM aspect “*simplicity as a driver for ad-hoc adaptations*” [34] and the ACM requirement “*understandable and adaptable by business users*” [37], knowledge workers without advanced technical skills need the ability to understand, follow, plan and improve cases. It is interesting to note that public administrations have a tendency to provide their knowledge workers with various powerful IT tools (courtesy of bulk licenses), but also to withhold adequate

training (due to cost-per-individual). It is therefore vital that IT systems in public administrations “keep it simple”, and provide user interfaces that are both intuitive and helpful. Furthermore, e-learning has proven a very effective factor for training users on new IT systems, as the lack of a classroom and trainer eliminates the cost-per-individual, which greatly simplifies the administrative overhead required to apply for training.

3.2.2 Coverage in CMMN

CMMN does not actually focus on ad-hoc processes (i.e. complete control over the model at run-time). Rather it supports the notion of pre-planned flexibility, by allowing case workers to choose at run-time from a set of discretionary items defined at design-time. While CMMN offers a powerful exchange format and notation for most elements of ACM (10 out of the 11 ACM requirements defined above are at least fulfilled to some extent), some aspects are not supported yet, and have to be provided by an ACM system. [6]

- **Ad-hoc changes:** Ad-hoc changes to a case can only be performed based on discretionary items defined at design-time. As discussed further below, an ACM system would have to provide the capability to perform ad-hoc changes during run-time, which have not been foreseen at design-time.
- **Visualisation:** Due to limiting support for run-time adaption to discretionary items, CMMN offers no method of visualizing other ad-hoc changes to a case. The lack of differentiation between items planned during design-time and items planned during run-time does not facilitate the identification of typical changes to the initial case plan. This complicates continuous improvement, which requires a thorough cross-case analysis of popular run-time changes performed by case workers.
- **Objectives:** CMMN lacks a proper mechanism for describing and visualising objectives. While milestones represent achievable targets and sentries can be used to model entry and exit conditions based on the case state, no explicit element is available to model the overall objectives of a case, stage or task.
- **Execution:** CMMN offers no guidance on how to represent ad-hoc adaptations to a case at run-time. In order to enhance a CMMN model with ad-hoc adaptations during run-time, a unified representation of design-time planning CMMN elements and run-time planning ACM elements is required.
- **Progress:** Visualising a case at run-time requires showing the progress of stages, tasks and milestones. As a design-time modelling notation, CMMN offers no such capabilities.

The existing coverage by CMMN of the 11 ACM requirements defined previously is described as follows. [6]

1. **Fulfilled (implicit):** The run-time adaption capabilities of CMMN are limited to discretionary items modelled at design-time. However tasks and stages can be skipped if not declared mandatory at design-time. This makes it possible to ignore optional sentries and tasks modelled during design-time, and instead add new sentries and tasks during run-time. While CMMN does not explicitly discuss further adaptations to the case model during run-time, it does not prohibit them either.
2. **Fulfilled (implicit):** CMMN models can describe both design-time case plans and run-time cases. This makes it possible to identify differences between the design-time and run-time representations of a case. There is however no visualisation for run-time

adaptions in CMMN, requiring an ACM system to provide a method for cross-case visualisation of changes to the initial case plan.

3. **Fulfilled (partial):** While CMMN allows for association of tasks and events with roles, it does not allow for association with specific users. An ACM system would have to provide its own implementation of the indirection layer between roles and users.
4. **Fulfilled (no visualisation):** While the metamodel allows for accurately tracking the progress of a case through completed tasks, stages and milestones, it offers no visualisation of the execution state of plan items.
5. **Fulfilled (no visualisation):** Roles can be assigned to specify responsibility for completing tasks or planning discretionary items at run-time. However CMMN offers no visualisation of responsibilities.
6. **Not fulfilled:** While the completion of tasks, stages and milestones can be measured, CMMN offers no explicit element for expressing case objectives. At best, the case file could hold a document outlining the case objectives, but such unstructured information is of little use for automated measurement of case progress.
7. **Fulfilled:** CMMN allows for modelling documents as well as arbitrary structured and unstructured information in the case file. It does not however enforce declaration of an explicit information type, which may negatively impact cross-case information exchange.
8. **Fulfilled:** CMMN defines stages as a recursive concept, which allows for hierarchical structuring of tasks using sub-stages.
9. **Fulfilled:** CMMN allows for the definition of complex dependencies between stages and tasks. Sentries and timer events can resolve expressions based on temporal logic and case state, surpassing the flexibility achieved through conditional sequence flows and events in BPMN.
10. **Fulfilled (implicit):** CMMN provides two elements for system interoperability. Process tasks allow for the execution of structured processes through an external execution engine for BPMN, XPD, or WS-BPEL, which in turn allows for the orchestration of external web services. The case file supports references to external document repositories through various standards including Content Management Interoperability Services (CMIS).
11. **Fulfilled (partial):** The similarities between BPMN and CMMN and the widespread acceptance of BPMN facilitate the adoption of CMMN for knowledge workers familiar with BPMN. However sentries allow for modelling highly complex dependencies between plan items, which may impede the legibility of such models.

3.3 Case Management Rationalisation

Based on a 2011 decision by the College of Commissioners (the collective decision-making body of the European Commission) to rationalise the existing IT systems, the Directorate-General (DG) for Competition was appointed to lead the Case Management Rationalisation (CMR) exercise. CMR covers the rationalisation domain of case management, aiming to deliver a new and efficient case management system for the DGs involved in this project. This system should cover all investigative and case-related activities used to enforce EU law, except for infringement procedures under Article 258 [38] of the Treaty on the Functioning of the European Union (TFEU).

The main objectives of CMR are defined as follows.

- Identify common case management activities of the DGs involved in the project

- Enable efficient sharing of case management data within and across DGs
- Improve effectiveness and efficiency of EU law enforcement
- Address immediate and long-term case management needs and opportunities
- Provide guidance to future case management activities
- Provide investment across all related functional areas

By managing information as digital assets and optimising back-office activities through the use of a case management system, DGs will be able to increase the efficiency of their case management activities. At the same time, by providing a consolidated IT system spanning a wide variety of case types, activities, contexts, and scopes, CMR will meet the IT rationalisation goal set forth by the College of Commissioners.

3.3.1 Lifecycle

The case management activities of the involved DGs span various case types and differ in their specific sequence of events. CMR has identified a shared lifecycle, covering the execution of a case as well as the management of documents and records related to it.

3.3.1.1 Case Management Lifecycle

The CMR case management lifecycle is divided into five stages, each of which is driven by the circumstances or events that occur during the stage itself.

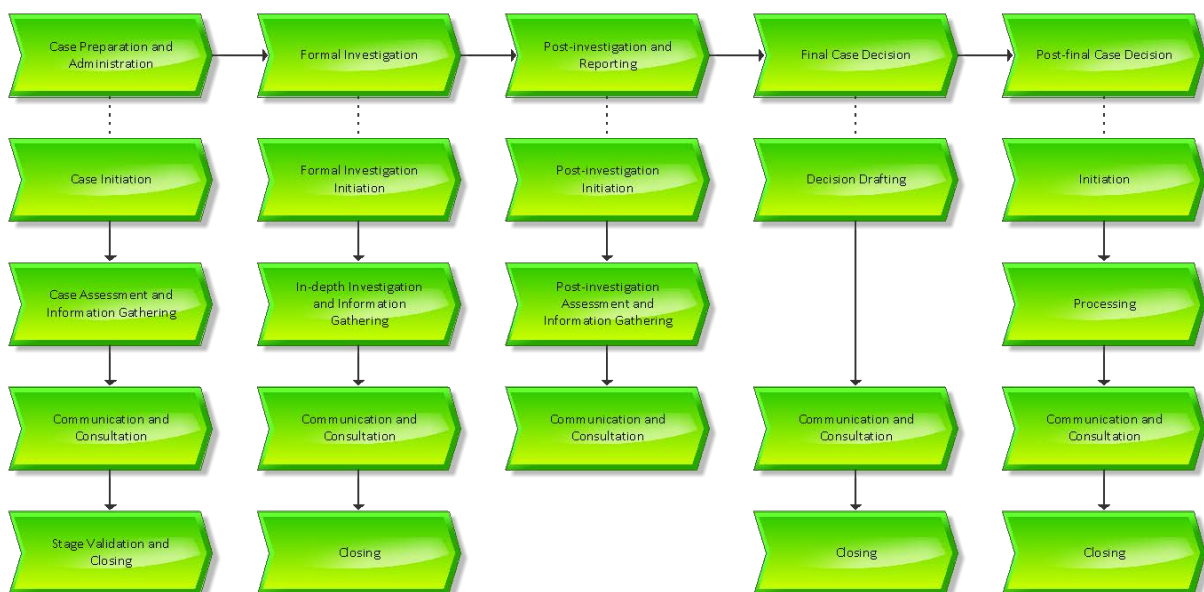


Figure 7 CMR Case Management Lifecycle

The progress of a case depends on the knowledge of the case team, and team members use the case management system to consult and collaborate both internally and externally, as well as create a complete record of discussions, positions and decisions related to the case.

1. **Case Preparation and Administration:** The first stage spans the preparation of resources and necessary administrative steps to start a case.
2. **Formal Investigation:** The second stage commences when a case is officially registered and resources are allocated. The case team can start investigations based on the relevant EU legislation, collect evidence and conduct required assessments.

3. **Post-investigation and Reporting:** The third stage includes the conclusion of the results of the formal investigation, the decision on the direction of the case, and the drafting of the final case decision.
4. **Final Case Decision:** The fourth stage covers the finalisation of the position of the case team, as well as the adoption of the final case decision.
5. **Post-final Case Decision:** The fifth and last stage groups the activities performed after the final decision and before the case is definitively closed.

The dynamic and knowledge-oriented aspects of a case further require each stage to be decomposed into four sub-stages, grouping specific activities into a generic sequence. It is interesting to note that, despite the generic nature of this sequence, not every sub-stage is in fact mandatory. Specifically, the third stage does not include a closing sub-stage, and the fourth stage does not include an administration and information gathering sub-stage.

1. **Initiation:** The first sub-stage includes activities required to start the case stage.
2. **Administration and Information Gathering:** The second sub-stage spans activities for arranging case resources and case planning, as well as assessing available information.
3. **Collaboration and Consultation:** The third sub-stage covers activities required to consult (e.g. share information) and collaborate (e.g. create documents) both within the case team, and with external parties involved in a case.
4. **Validation and Closing:** The fourth and final sub-stage groups activities required to validate the result and close a case stage.

3.3.1.2 Document and Record Management Lifecycle

The CMR document and record management lifecycle is divided into four stages.

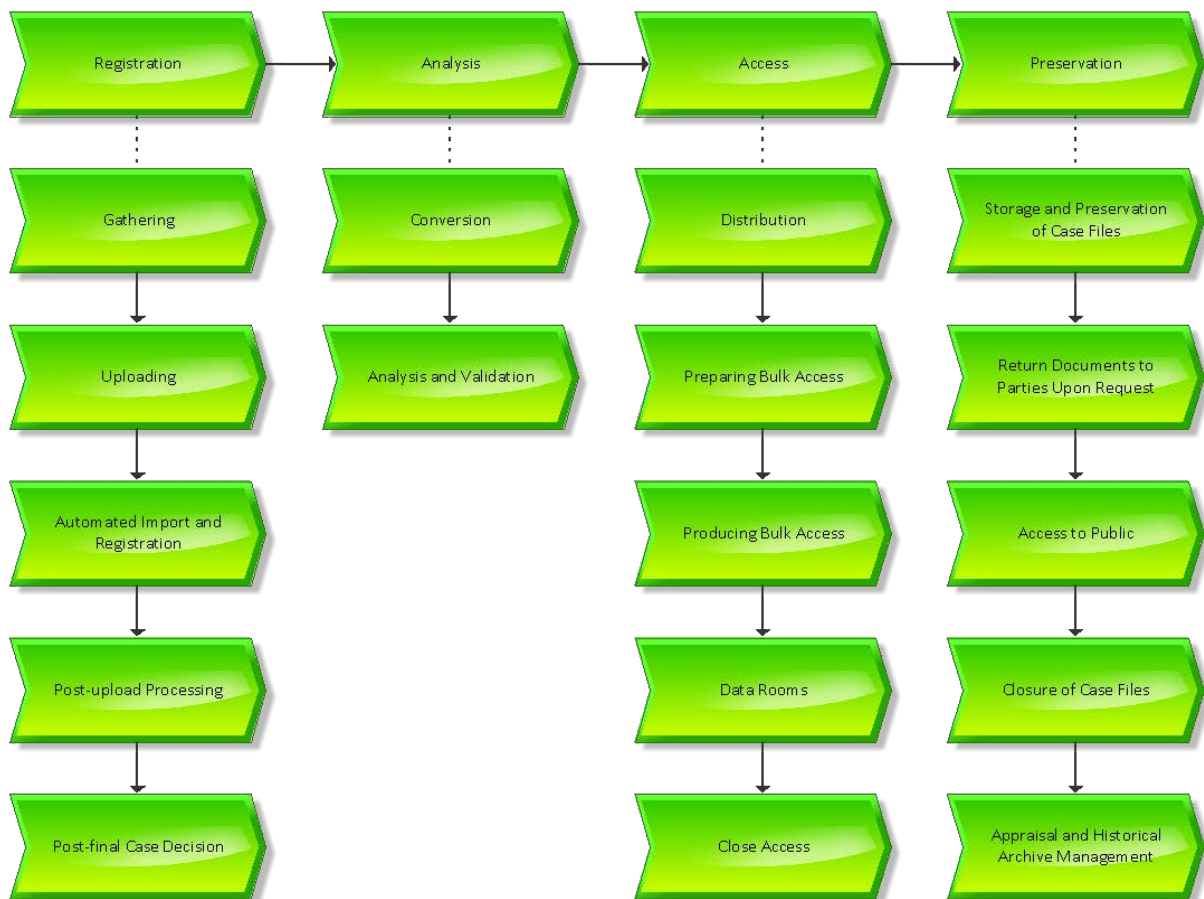


Figure 8 CMR Document and Record Management Lifecycle

The four stages cover the activities occurring over the entire lifecycle of a document or record, from initial creation or capture to final destruction.

1. **Registration:** The first stage groups activities required to officially and formally register case documents.
2. **Analysis:** The second stage covers activities involved in processing and validating case documents.
3. **Access:** The third stage includes activities related to sharing and distributing case documents requested by any party involved in a case.
4. **Preservation:** The fourth and final stage spans activities required to retain and preserve case documents and records for a legally specified period of time, as well as their eventual destruction.

Each stage is further decomposed into a non-generic sequence of sub-stages, similar to the decomposition of case management stages.

3.3.2 Activities

A detailed review of the activities identified by CMR would go beyond the scope of this thesis, but let us take a quick look at the list to gain an impression of the various functionalities a case management system for public administrations has to provide.

3.3.2.1 Case Management Activities

A total of 46 case management activities have been identified by CMR. 7 of these are common activities, i.e. they do not correlate to a specific stage. Instead they cover the underlying functionality of a case management system.

Activity	Stage	Sub-stage
*Create case file	1	1
*Receive initial case documents	1	1
*Register initial case documents	1	1
*Translate initial case documents	1	1
*Plan case calendar	1	1
*Assign case team	1	1
*Assign tasks and activities to case team	1	1
*Assess initial case documents	1	2
Collaborate and consult on case preparation	1	3
Validate and close case preparation	1	4
Open formal investigation	2	1
Plan official case deadlines in case calendar	2	1
Update case file	2	1
Review case file	2	1
Register newly created or received case documents	2	1
Review case team assignment	2	1
Assess formal investigation documents received	2	2
*Prepare request for information	2	2
*Analyse and assess gathered information	2	2
Collaborate and consult on formal investigation	2	3
Validate and close formal investigation	2	4
Validate result of formal investigation	3	1
Consider case resolution	3	1
Prepare case decision	3	1
Prepare access to case file	3	1
Prepare request for information	3	2
Prepare market investigation	3	2

Activity	Stage	Sub-stage
Analyse and assess gathered information	3	2
Collaborate and communicate on post-investigation	3	3
*Prepare and draft final case decision	4	1
Collaborate and consult on final case decision	4	3
Translate final case decision summary	4	4
*Validate and adopt final case decision	4	4
Notify involved parties of final case decision	5	1
Publish final case decision	5	1
Assess documents received from legal service	5	2
Prepare for court hearing	5	2
Collaborate and consult on post-final case decision	5	3
*Monitor implementation of final case decision	5	4
Workflow Administration	-	-
User Role Management	-	-
Electronic Approval	-	-
Document Drafting	-	-
Activity Management	-	-
Task Management	-	-
Event Management	-	-

Table 2 CMR Case Management Activities

It should be noted that, while most activities are unique to their respective stage, some (e.g. collaborate and consult) are repetitive patterns occurring in multiple or even all stages. Furthermore, 13 activities prefixed with an asterisk (*) in the table above, plus the 7 common activities as well as the recurring pattern of collaborate and consult have been identified as relevant to so-called horizontal cases. Unlike the well-defined cases of competition law, such cases do not follow clear procedural rules, and require a considerable amount of flexibility and ad-hoc process definition. They do however share the need for document and record management, as well as collaboration, consultation and reporting.

3.3.2.2 Document and Record Management Activities

A total of 40 document and record management activities have been identified by CMR. 10 of these are common activities, i.e. they do not correlate to a specific stage. Instead they cover the underlying functionality of a document and record management system.

Activity	Stage	Sub-stage
----------	-------	-----------

Activity	Stage	Sub-stage
Review initial physical document	1	1
Create case folder	1	1
Scan paper documents	1	1
Upload electronic documents, e-mails and collections	1	2
Add registration metadata	1	2
Perform manual post-upload processing	1	3
Automate import and registration from corporate tools	1	4
Register, link and file documents	1	5
Acknowledge and notify documents	1	5
Perform optical and intelligent character recognition	2	1
Convert to PDF format	2	1
Compress document	2	1
Validate metadata	2	2
Export to corporate record management system	3	1
Distribute to website or print publication	3	1
Print on demand	3	1
Syndicate and transform content	3	1
Prepare and check confidentiality	3	2
Request and review non-confidential version	3	2
Produce bulk access version	3	3
Manage data rooms	3	4
Close access to file requests	3	5
Preserve document	4	1
Return document to party involved in case	4	2
Provide public access to document	4	3
Close case file temporarily or definitively	4	4
Transfer case file to temporary archive	4	4
Review and sample case file	4	5
Transfer case file to historical archive	4	5
Eliminate case file from historical archive	4	5

Activity	Stage	Sub-stage
Automatic classification of documents	-	-
Support for multi-lingual metadata	-	-
Viewing and rendition of documents	-	-
Support for multi-lingual full-text search	-	-
Quarantine environment	-	-
Digital and information rights management	-	-
Metadata model definition management	-	-
Synchronisation and retention schedule management	-	-
Classification schema management	-	-
File plan management	-	-

Table 3 CMR Document and Record Management Activities

It is interesting to note that – unlike automated case management – the concept of automated document and record management is already well-established in EU public administration. While document and record management are certainly a core feature of any case management system, the unusually high ratio of document and record management activities to case management activities (40:46 in the classification shown above) is likely the result of a deeper understanding of the needs and opportunities in the domain of document and record management.

3.3.3 Building Blocks

Based on the case, document, and record management activities introduced above, CMR has identified 35 content clusters, grouped into 8 building blocks.

1. **Case Management (CM):** Covering all case-related elements, including initiation, planning, resource and task allocation, operational reporting, and portfolio management.
 - Resource Management (RM)
 - Task Management (TM)
 - Calendar Management (CAL)
 - Case Support (CS)
 - Document Template Management (DTM)
 - Reporting and Dashboard (RPT)
2. **Case Handling (CH):** Covering all case activities, including decision drafting and contact management, as well as assessment, analysis, and preparation of the legally required output based on formally defined procedures and legal rules.
 - Document Negotiation (DN)
 - Case Progress (CP)
 - Workflow Execution (WE)
 - Evidence Management (EM)
 - Reporting (RPT)

3. **Document Management (DM)**: Capturing, processing, converting and distributing collections of documents.
 - Registration (REG)
 - Scanning (SCN)
 - E-mail Handling (EH)
 - Import / Export (IMEX)
 - Browsing, Searching and Viewing (BSV)
 - Printing and e-Distribution (PD)
 - Conversion Services (CS)
 - Reporting (RPT)
4. **Records Management (RM)**: Preserving and controlling information that has an evidential value to the organisation.
 - Manage (MG)
 - Preserve (PR)
 - Report (RPT)
5. **Collaboration (CO)**: Consultation, interaction, collaboration, and discussion with team members and outside parties to reach a common objective.
 - Document-related Collaboration (DC)
 - Case-team and Inter-service Collaboration (TS)
 - Reporting (RPT)
6. **Orchestration (OR)**: Managing business rules, as well as pre-defined structures and configuration templates.
 - Workflow Template Definition (WTD)
 - Semi-structured Task Selection (WE)
 - Business Rules, Alert and Notification Management (BR)
7. **Administration and Configuration (ADCO)**: Management of individual building blocks.
 - Organisation Management (CM)
 - User and Role Management (UM)
8. **Tools (TO)**: Non-core applications and services facilitating case, document and record management.
 - Translation Tool (TT)
 - Access to File Tool (A2F)
 - Investigative Tool (INV)
 - Quarantine Tool (QUA)
 - Off-line Usage (OU)

3.4 Requirements Selection for Prototype

Given that ACM aims at providing users with all the tools required to perform knowledge work, the extensive list of requirements discussed above should come as no big surprise. To limit the scope of the prototype used in this thesis, the following core requirements have been selected.

Requirement	Elements of ACM	Mapping CMMN to ACM	CMR Content Clusters and Building Blocks
Interoperability	Decision Making, Content, Security	Content Integration,	Document Management, Collaboration

Requirement	Elements of ACM	Mapping CMMN to ACM	CMR Content Clusters and Building Blocks
		Interoperability	
Process Execution	Case Templates	Case Templates, Run-time Flexibility, Hierarchical Task Structure	Workflow Template Definition, Workflow Execution, Semi-structured Task Selection, Task Management, Calendar Management
Decision Making	Decision Making	Temporal-logic Dependencies	Business Rules Management
Content Management	Content, Record Management	Content Integration	Document Template Management, Document Management, Record Management
Identity and Access Management	Authentication, Authorisation	Responsibilities, Roles	Administration and Configuration
Accountability and Transparency	Accountability, Transparency	Responsibilities, Cross-case Improvement	Resource Management, Reporting and Dashboard

Table 4 Mapping of selected ACM requirements

The following ACM requirements will not be further discussed: encryption, confidentiality and authenticity as defined by Pucher; case progress, objectives, and legibility and usability as defined by Kurz et al.; and numerous content clusters and building blocks defined by CMR, which are not related to the unpredictability of knowledge work.

3.4.1 Interoperability

Single source of information and re-use of existing functionalities are core concepts of modern IT systems, and ensuring interoperability is essential to the successful use of open source software in large-scale IT applications. While vendor-based technology stacks attempt to cover the complete range of functionalities required by an IT system, open source-based solutions have to integrate a wide range of different components, each providing a piece of the puzzle.

As stated by Pucher and discussed in chapter 3.1, ACM has to provide a universal approach integrating various knowledge silos into a unified and meaningful view. In chapter 3.2 Kurz et al. recommend the re-use of existing standards and a focus on integration with other systems. And the CMR building blocks presented in chapter 3.3 include Document Management, Record Management and Collaboration, all of which directly depend on the interaction with external systems.

The ACM prototype must provide interoperability by re-using existing standards that facilitate interaction with external IT systems, to both provide and consume data and services.

3.4.2 Process Execution

When the need for ad-hoc adaptations of a process instance first arose, leading vendors of business process execution engines claimed to support dynamic processes by implementing an ad hoc task as a completely unstructured container for tasks defined at run-time. Clearly, this approach falls short of supporting the needs of case workers, who deal with shades of predictability, rather than strictly predictable or unpredictable processes.

In chapter 3.1.2 Pucher considers the management of case templates that go beyond traditional BPMN models. As discussed in chapter 3.2, Kurz et al. support Pucher's requirement for case templates that depict the run-time changes made by knowledge workers, as well as stating the need for an ability to adapt all parts of a case at run-time, and recommending a recursive hierarchy of tasks and processes. The CMR building blocks presented in chapter 3.3 include the management of workflow definitions and semi-structured tasks, while also addressing a practical aspect of case management in public administrations – case planning through a case calendar.

The ACM prototype should demonstrate the level of unpredictability supported by the chosen process execution engine, and investigate the integration of case template management capabilities.

3.4.3 Decision Making

Decision making in process execution takes place on various levels. Some rules apply organisation-wide, and are independent of the specific context. Other rules apply in the context of an IT system, but are not specific to the various processes implemented by the IT system. Most rules however depend on the specific context of a process, i.e. the process state, which changes over time.

In chapter 3.1.3 Pucher advocates the importance of the process state for decision making, and suggests the formulation of case goals as rules. Kurz et al. propose the definition of rules based on temporal logic to define the order in which case activities are completed (see chapter 3.2.1). The CMR content cluster Business Rules Management is part of the Orchestration building block, suggesting that rules should be managed centrally and re-used in specific cases.

The ACM prototype should provide the capability to make decisions both based on the process state and based on the application state, and investigate the non-sequential ordering of case activities.

3.4.4 Content Management

Knowledge work is content-driven, i.e. it relies on explicit knowledge for both input and output. In a business domain like trade negotiations, documents like the negotiation directives set out at the start, or the final text of the trade agreement adopted at the end, are the primary artefacts around which the entire case revolves. The various representations of content (e.g. documents, records, emails, semi-structured information or structured data) have to be integrated by an ACM system.

Pucher states in chapter 3.1.4, that process and content are meaningless without each other, and elaborates on the need of storing historical content as records. In chapter 3.2.1 Kurz et al. state the data- and document-centric nature of ACM, and demand tight integration with standard document and record management solutions. The CMR building blocks Document

Management and Record Management reinforce the importance of content, while the content cluster Document Template Management addresses the need to provide knowledge workers with more than just a blank sheet of paper.

The ACM prototype must demonstrate its ability to associate content with process elements, and investigate the integration with external content repositories.

3.4.5 Identity and Access Management

Identity and Access Management (IAM) enables the right individuals to access the right resources at the right times and for the right reasons, by addressing the need to ensure appropriate access to resources across increasingly heterogeneous technology environments, and to meet increasingly rigorous compliance requirements. [39]

While standard BPM supports the concept of swimlanes to define responsibility, the only process element intended for human interaction is the user task. Many other process entities exist however, and vendor-based BPM solutions tend to rely on functional authorisation, assuming that any user who might want to perform an advanced action (e.g. suspending a process instance, or updating a process definition) must be an administrator with an appropriate functional role. This prevents knowledge workers from taking control of their own cases, and a novel approach is required to enforce adequate access control for all interactive process entities.

Pucher states in chapter 3.1.5 that all acting users must be authenticated and authorised. In chapter 3.2.1 Kurz et al. likewise state that every action on a process element must be associated with a candidate user or group, and that access rights must be adaptable at run-time. The CMR building block Administration and Configuration includes the content clusters Organisation Management (i.e. managing hierarchical groups) and User and Role Management (i.e. managing users and functional roles), and relies on the tenancy concept (i.e. restricting access to information based on the organisation a user belongs to) to manage case records from various Directorate-Generals (DG) in the same IT system.

The ACM prototype must demonstrate access control based on tenants, users, groups and roles, and investigate how unauthorised actions on process entities can be prevented.

3.4.6 Accountability and Transparency

While IAM prevents unauthorised access to application components and business objects, accountability ensures that records are kept of any such access. This allows organisations to perform an audit to evaluate if an IT system is operating effectively, and in case of a security incident, enables the identification of authorised users who performed malicious actions.

“An information technology audit, or information systems audit, is an examination of the controls within an Information technology (IT) infrastructure. An IT audit is the process of collecting and evaluating evidence of an organization's information systems, practices, and operations. The evaluation of obtained evidence determines if the information systems are safeguarding assets, maintaining data integrity, and operating effectively to achieve the organization's goals or objectives.” [40]

Auditing is required in most large organisations as an instrument to confirm that the actions taken by its members conform to the rules and regulations of the organisation. The minimum information that has to be captured by the case management system is who (i.e. user or group) did what (i.e. functionality), when (i.e. date and time) and where (i.e. on which

business object). Recording the why (i.e. justification) is optional, as it would require the user to enter a comment when taking any action in the case management system.

“Transparency is operating in such a way that it is easy for others to see what actions are performed.”[41]

Providing non-confidential information in a timely manner serves the organisational goals of openness, accountability, and generation of trust, and enables stakeholders to stay up-to-date on the actions performed by a given organisation. Transparency allows knowledge workers to continuously improve their case processes, and managers to have an overview of the current state of cases under their responsibility, and offers users of a case management system a direct and efficient access to all relevant information required to perform their tasks.

Keeping a historical record of past activities allows knowledge workers to analyse their own actions on past cases, and is essential for the analysis and redesign of processes. When knowledge workers plan a new case, they draw on their experience of past cases. Recording and visualising this experience allows an organisation to capture explicit knowledge from every case executed in the case management system. This in turn supports both knowledge workers and the organisation in continuously improving their processes, by keeping track of past actions and their consequences. Reports based on the information collected are an essential instrument to keep stakeholders in the loop on the progress of a case, and to provide upper management with an overview of the current status of all cases.

“A dashboard should always incite emotion or action. If it’s not clear to your audience why the data is relevant, take feedback and change the dashboard. It may take an iteration or two to properly tune your dashboard to your team and stakeholders.”[42]

Browsing information in its natural state works well if you are looking for detailed information on all entities. But automated systems – and especially case management systems – produce an enormous amount of information, of which only a small part may be relevant to a specific user at a given point in time. Modern user interfaces therefore feature dashboards, which aggregate information relevant to a specific user, and present this information in a customizable way.

The ACM prototype should: keep a historical record of the process state, as well as of all business objects managed outside the execution engine; present the information collected during process execution; and provide both the UI components and data retrieval capabilities required to build a personalised dashboard.

4 Practical Application of Case Management

Based on two projects recently completed at the European Commission (EC), this chapter discusses lessons learned from the practical application of case management in public administration. Negotiation Support Tool (NEST), a prototype for a business process-oriented case management system to support trade negotiations, was developed at the Directorate-General for Trade (DG Trade). NEST assists negotiators in tasks like document, group and calendar management, applying an adaptive case management approach to address the unpredictability of workflows in knowledge work. Customs Decisions (CD), a service- and business process-oriented workflow management system to support the Union Customs Code (UCC) legislation will be released by the Directorate-General for Taxation and Customs Union (DG TAXUD) on October 1, 2017. CD connects European traders with customs officers using executable business processes to manage the lifecycle of UCC

authorisations, as well as Member States' IT systems with DG TAXUD's central SOA infrastructure, using a hybrid approach allowing each Member State to choose between consuming nationally implemented or centrally provided web services.

4.1 Stakeholders

Before discussing the development of a case management system, we first have to understand the context of developing IT systems for the public administration of the European Union (EU). Projects at the EC follow the Project Management Methodology (PM²), a customised project management methodology, which is tailored towards the environment and needs of EU institutions and projects, and encompasses the complete lifecycle of a project. PM² defines project stakeholders as follows.

Project stakeholders are people (or groups) who can affect or be affected by the activities carried out during a project's lifecycle and/or by the project's output(s) and outcome(s). Stakeholders can be directly involved in a project's work, can be members of other internal organisations, or can even be external to the organisation (e.g. suppliers, users, general public, etc.). [43]

Simply put, a stakeholder is anyone involved in or affected by a project, and stakeholder management is critical to the successful completion of a project. Stakeholders of an IT project at the EC can be both outside of the organisation (e.g. EU Member States, EU citizens, other EU institutions, and external IT contractors), or inside (e.g. upper management, policy or business units, IT units of other DGs providing or consuming data or services to or from the project, and user representatives). Each stakeholder has needs – often competing with or contradicting other stakeholders' interests – and various platforms exist for stakeholders to express their requirements and consent to the proposed solution.

4.1.1 External Stakeholders

An IT project developed at the EC can have various external stakeholders. Some projects like NEST have no external stakeholders, as both end users (trade negotiators) and IT contractors (intra-muros consultants) are within the organisation. Other projects like CD have a wide range of external stakeholders, as both end users (customs officials and economic operators from EU Member States) and IT contractors (external companies) are outside the organisation.

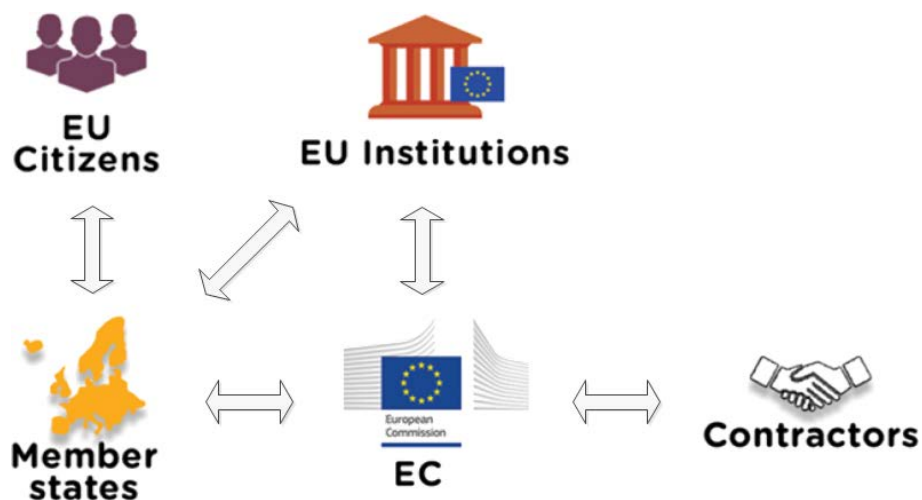


Figure 9 External Stakeholders of IT Projects at the EC

EU Member States can be directly affected by IT projects developed at the EC. Be it as providers or consumers of data, or (in the case of Trans-European IT projects) as providers or consumers of services and user interfaces. DG TAXUD provides various platforms to involve Member States in the design phase of Trans-European IT systems. Working groups allow customs representatives from some EU Member States to provide feedback on the current state of the application design, and offer suggestions for refining the requirements towards the needs of national customs administrations. Review cycles offer each Member State the possibility to comment on the working version of the IT specifications, and provide change requests based on their specific needs. Finally, the output of work performed by the EC is validated in recurring meetings with representatives from all Member States, e.g. the Electronic Customs Coordination Group (ECCG) which formally accepts the requirements produced by the business unit, and the IT Systems Development Group (ITSDG) which formally accepts the IT specifications produced by the IT unit.

EU Citizens' interests are usually not directly represented in an IT project, rather EU Member States act as the representative of their citizens interests. Similarly, other EU institutions are usually not directly involved, though they may have a significant indirect effect, e.g. the EU parliament which has to formally accept the UCC legislation, which serves as the legal basis of the CD project. Any delays incurred in the formal acceptance of the legal text, can have a severe impact on the parallel planning of an IT project required to support the legislation.

External contractors are a powerful factor in the development of IT systems at the EC. Relying on external contractors introduces a significant dependency (and thus risk), and some DGs like DG Trade prefer to develop customised solutions in-house. However, this also sets limits in regard to project size and technological expertise, and other DGs like DG TAXUD rely on external companies which hold framework contracts to perform software development, quality assurance and service management. Working with external contractors limits the influence one can exert over these areas to contractually agreed terms, which in turn puts an emphasis on contract management as the primary tool for influencing the success of an IT project.

4.1.2 Internal Stakeholders

PM² divides project organisation into two sides, the requestor expresses a need and the provider supplies a solution. In the context of IT projects, the need could be an IT system to support a piece of legislation drafted by a policy unit acting as requestor. The solution is provided by either the DGs own IT unit, or the central DG for Informatics (DIGIT), and can consist of e.g. granting access to an existing IT system, adapting an existing IT system to meet the specific needs of the project, or developing a new IT system.

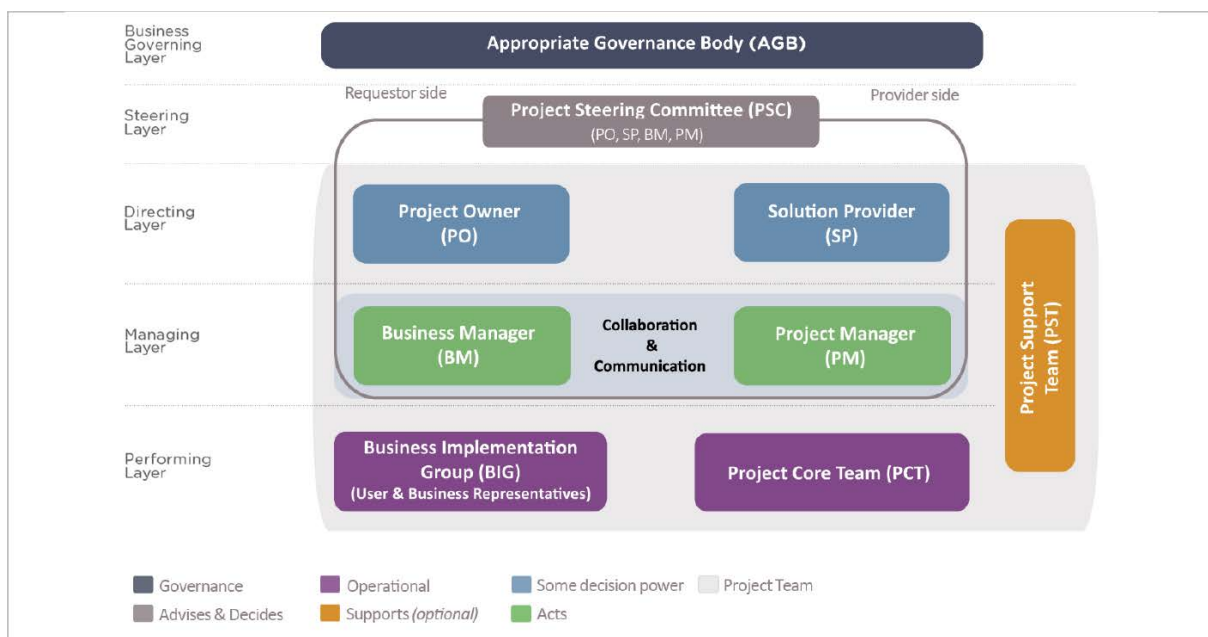


Figure 10 PM² Project Organisation [43]

From the point of view of an IT project at the EC, the upper management consists of the Project Steering Committee (PSC), which acts as the key decision body of a project, all the way up to the Director and in theory even the Director-General (though in practice an involvement at that level is exceedingly rare). Each level of management may have its own needs, based on a much bigger picture than a single IT project.

The duality of requestor and provider is the source of several typical issues between policy and business units. While both sides have a vested interest in the success of the project, at the same time they also want to make sure that in case of failure, the blame is attributed to the other side. Once an IT system is up and running, responsibility for the continued maintenance relies both on the requestor (e.g. performing content updates in case of legislative changes) and the provider (e.g. performing software updates and maintaining the IT infrastructure), with either side frustrated by failures like delays by the requestor in the provision of requirements or inability by the provider to fulfil service level agreements. Finally, the distinct differences between non-technical requestors like policy units, and technical providers like IT units create great challenges in communication, as both sides carry a very different mindset, and indeed speak a different language.

For this reason the organisational structure of many DGs includes a business unit, which is responsible for capturing the business processes of policy units. The business unit then acts as requestor for IT projects, providing requirements to the IT unit based on the captured

business processes. The existence of a mediating layer between policy and IT units is based on the ideas taught in the academic field of business informatics, but in the context of public administration this approach yields a possible detrimental effect. Due to the high level of legal knowledge required to understand legislation, it is nearly impossible to find candidates who possess the required technical background to translate between requestors and providers. A continued need for business analysts to perform the actual translation of business needs on the provider side however results in what is effectively an added layer of potential misdirection between requestor and provider.

IT Units of other DGs can be stakeholders as well. Many DGs provide specialised IT solutions to other DGs, and DIGIT acts as a central provider of horizontal IT services. The resulting issues of this interaction are similar to the duality of requestor and provider. Consuming an external IT service creates a dependency and yields the risk of planning delays or underperforming service levels. Likewise, providing an IT service to external consumers creates a dependency by introducing external goals, which may conflict with the own needs of the provider.

An often under-represented stakeholder group are the user representatives, who in the context of ACM represent the knowledge workers that use the provided IT system to perform case work. While the need for usable and accessible IT systems is well understood at the EC, in practice legally mandated deadlines require the existence of an IT system.

4.2 Predictability

Identifying the level of predictability in a business domain, and choosing the appropriate case management approach is the first issue in developing a new case management system. Let us consider two extreme examples: the fully predictable and automatable domain of corporate document scanning services; and the highly unpredictable and non-automatable domain of project functional mailboxes.

DIGIT provides corporate document scanning services using multi-function printers to scan paper documents, convert them to PDF, and send the result to the user's corporate email address. The only human interaction is placing the document on the scanner and identifying yourself to the system, everything else follows a fully automated process. There is no need for flexibility, as the simplicity of the process allows for all possible events and activities to be known in advance.

Functional mailboxes on the other hand tend to be a physical manifestation of chaos theory. With every stakeholder addressing any remotely related communication to the formal point of contact of a project, functional mailboxes are enormous collections of unstructured information. From repetitive robomails announcing yet another updated ticket from the development team, to panicked requests for immediate assistance from the service management team, anything is possible. Evaluating, tracking and resolving all possible issues cannot be automated, as neither the complete set of problems nor the complete set of solutions is known in advance.

Most business domains of the EC fall somewhere in-between these two extremes, and identifying the level of predictability and choosing the appropriate case management approach for a specific business domain is critical to the success of a project. The correct balance allows knowledge workers the flexibility required to achieve optimal results, while retaining the necessary amount of control for decision makers.

4.2.1 High Predictability

Customs duties amount for 15% of total EU revenue [44], making the customs union a key area of interest for both Member States and EU Institutions. DG TAXUD manages some of the largest IT projects at the EC, maintains its own data centre, and has developed its own methodologies for business process modelling, i.e. EU Customs BPM Methodology [45], and software development, i.e. TAXUD's Electronic Management of Projects Online (TEMPO) and Software Development Life Cycle (SDLC). The work of Member States' customs officials is facilitated by several Trans-European IT systems, based on legislation drafted by the EC and formally accepted by other EU institutions as well as the Member States.

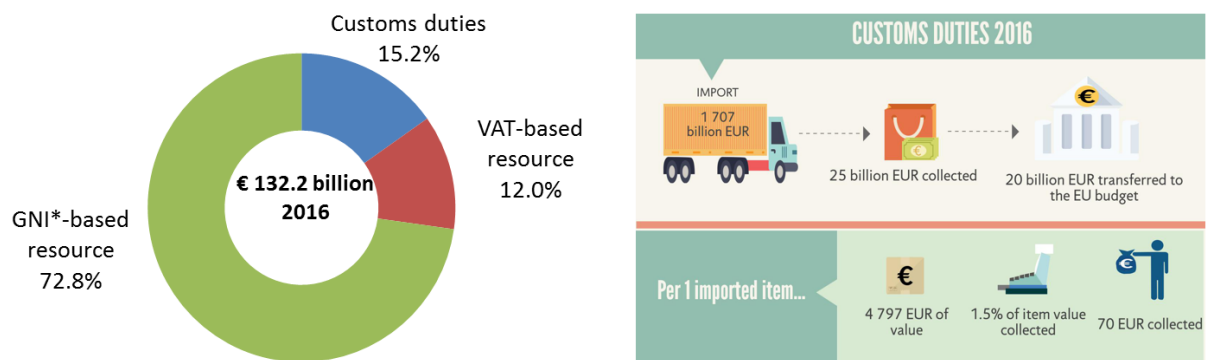


Figure 11 EU Revenue and Customs Duties [44]

The customs union is an example of a domain with high predictability. The direct involvement of Member States and the magnitude of the user base (CD addresses the needs of 45.000 national customs officials and 4.500.000 registered economic operators active in the EU) require prior agreement on and strict control over the lifecycle of customs decisions. All Member States must respect the same legal framework, and all economic operators must be treated equally. The legal act, the resulting business processes and the supporting IT systems must be planned many years in advance. Change requests are only possible with the agreement of all key stakeholders, and can take up to 2 years to implement. It's an environment that yields little room for flexibility, and is therefore well suited for structured process execution.

4.2.2 Low Predictability

The EU manages trade relations with countries outside of the single market through its trade policy, which is an exclusive power of the EU institutions (i.e. individual Member States are not allowed to negotiate trade agreements). The EC negotiates trade agreements on behalf of the EU, while the EP and Council have to approve the final agreement. The EU currently has 32 trade agreements in place, 45 trade agreements partly in place, and 24 trade agreements being negotiated or updated. [46]

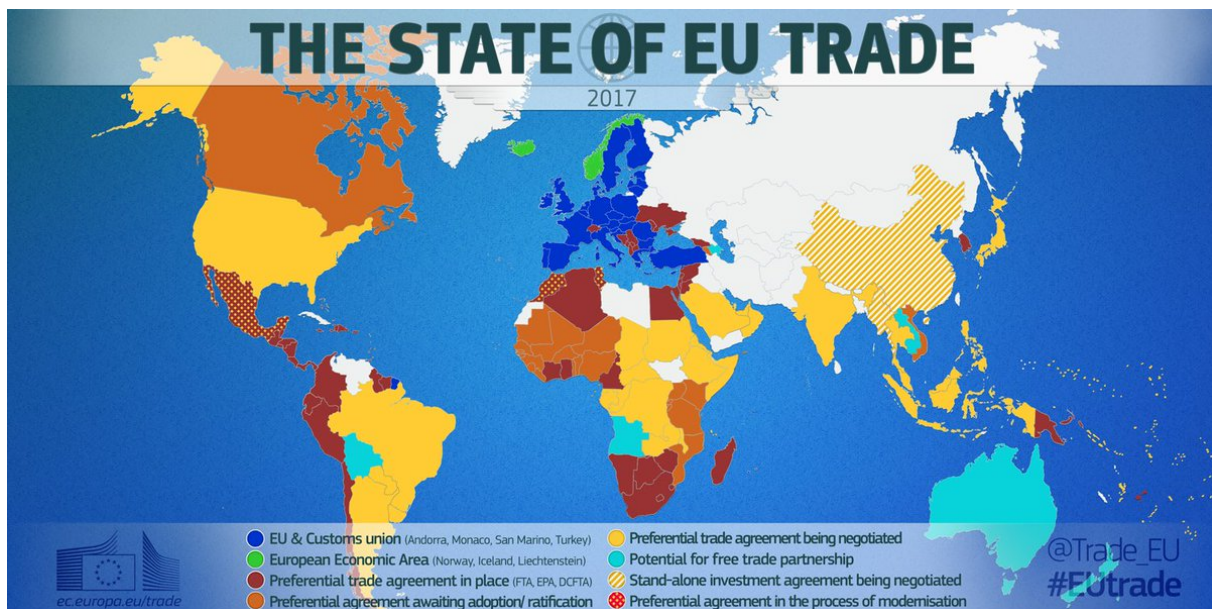


Figure 12 State of EU Trade 2017 [47]

The EC, the EP and the Council are in regular contact regarding trade policy. When trade negotiations with a certain country are proposed, a public consultation and impact assessment are held. The EC then starts an informal dialogue with that country, in the form of a scoping exercise which assesses whether a deal would be feasible. If the EC requests formal authorisation to open negotiations, the Council defines the general objectives of the agreement as negotiating directives, and authorises the EC to negotiate on behalf of the EU. [48]

While trade negotiations follow a general pattern (i.e. preparation, negotiation, conclusion, application), they offer a lot of room for flexibility. The negotiating directives set out goals, but they do not prescribe a specific way of reaching those goals, instead relying on the skills and experience of trade negotiators to achieve the desired outcome. Trade negotiations are therefore not well suited for structured process execution, and can benefit greatly from an adaptive approach to case management.

5 Prototype

Marin et al. state that predominant workflow management solutions are too rigid and provide no means to deal with unpredictable situations. [49] The ACM prototype developed for this thesis in late 2014 is based on an open source technology stack, thus allowing for customisation of the process execution engine itself. The customisation approach chosen for the ACM prototype falls short of a full fork, instead trying to minimize the impact by deploying only customised resources over a standard version of Activiti. Though it is a labor-intensive manual process, this approach allows for updating the customised resources based on newer versions of Activiti.

This chapter presents the architecture of the ACM prototype, which is largely based on the work done on the Negotiation Support Tool (NEST) prototype in 2013. However the results of the research presented in chapter 2, as well as the lessons learned from other case management projects, conducted at the European Commission (EC) in 2014 and discussed

in chapter 4, have led to several adaptations. Specifically, the ACM prototype explores the execution of unpredictable processes (discussed in chapter 7.2), and access control to process entities based on a collaborator lifecycle (discussed in chapter 7.5).

5.1 Web Application

Developing a web application initially requires 3 choices to be made: the programming language in which the application is written; an application server on which the web application is deployed; and an application framework which supports the development of web applications. With each choice taken, the field of candidates for the next choice may be reduced, as e.g. Microsoft's .NET application framework usually depends on C# as programming language and Internet Information Services as web server. [50]

5.1.1 Programming Language

The TIOBE Programming Community Index measures the popularity of programming languages by counting query results on the 25 most popular search engines [51]. Like other indices such as PYPL Popularity of Programming Language [52] and IEEE Spectrum's 2017 ranking [53], it finds Java to be one of the most popular programming languages.

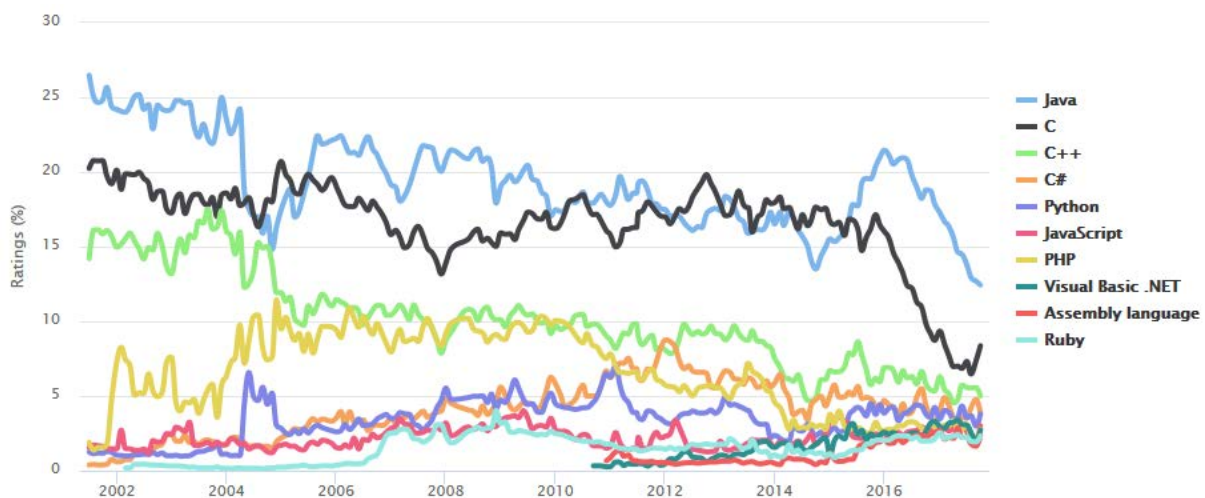


Figure 13 TIOBE Programming Community Index [51]

Java is a general-purpose, concurrent, class-based and object-oriented language, which is compiled to bytecode and executed on a Java Virtual Machine (JVM), making it effectively platform-independent (i.e. Java applications can be run on both Windows and Linux servers). [54] Java's wide acceptance results in a large and active open source community, and facilitates the search for skilled developers, making it an excellent choice for an open source-based solution.

5.1.2 Application Server

DZone publishes an annual report on the market distribution of Java application servers, based on data collected from 1600 JVM deployments utilizing Software-as-a-Service installations of Plumb for performance monitoring. [55] Over the past 4 years, Tomcat has consistently ranked as the most popular Java application server, achieving a 63.8% market share in the 2017 report.

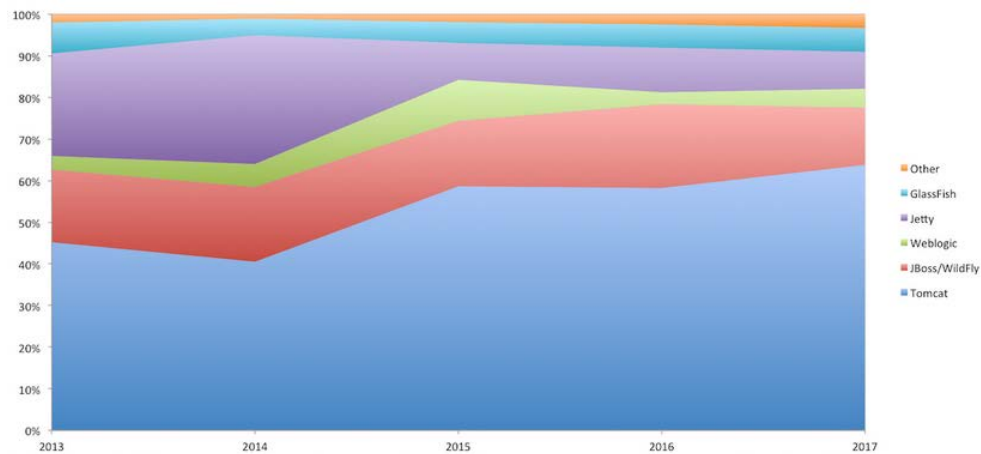


Figure 14 Java application server market distribution [55]

Tomcat is an open source Java servlet container, developed by the Apache Software Foundation, which implements several Java Enterprise Edition specifications including Servlet, JavaServer Pages (JSP) and Unified Expression Language (EL). Different versions of Tomcat support different versions of Java, and for the ACM prototype the combination of Java 8 and Tomcat 8 (including Servlet 3.1, JSP 2.3 and EL 3.0) was chosen. [56]

5.1.3 Application Framework

The Java Tools and Technologies Landscape Report 2016 published by RebelLabs analysed the tools and technologies in use by Java developers. [57] It is based on a survey of 2000 developers, architects and team leads, as well as other IT project stakeholders, working for mostly enterprise and mid-sized companies. The report finds that Spring MVC has seen a steady rise in popularity, and has consistently been the most popular Java web application framework since 2012.

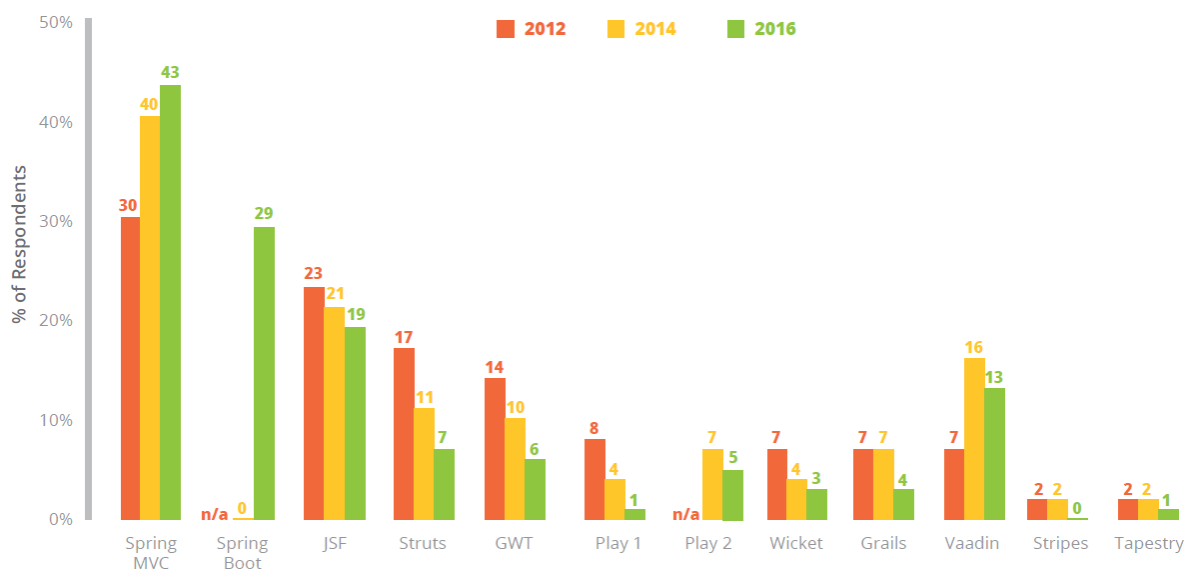


Figure 15 Web application framework usage since 2012 [57]

Spring MVC is an HTTP- and servlet-based web application framework that is part of the Spring framework, which also provides various other modules used by the ACM prototype, such as Spring Context and Spring Security. [58] Spring web applications are servlet-based, and are deployed on a web server by providing a deployment descriptor, which defines container options, security and configuration settings specific to the application.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<WEB-APP VERSION="3.1" XMLNS="HTTP://XMLNS.JCP.ORG/XML/NS/JAVAAE"
  XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
  XSI:SCHEMALOCATION="HTTP://XMLNS.JCP.ORG/XML/NS/JAVAAE
HTTP://XMLNS.JCP.ORG/XML/NS/JAVAAE/WEB-APP_3_1.XSD">
  <!-- THE DEFINITION OF THE ROOT SPRING CONTAINER SHARED BY ALL SERVLETS AND
FILTERS -->
  <CONTEXT-PARAM>
    <PARAM-NAME>CONTEXTCONFIGLOCATION</PARAM-NAME>
    <PARAM-VALUE>
      CLASSPATH:/SPRING/ROOT-CONTEXT.XML
      CLASSPATH:/SPRING/SECURITY-CONTEXT.XML
    </PARAM-VALUE>
  </CONTEXT-PARAM>
  <!-- CREATES THE SPRING CONTAINER SHARED BY ALL SERVLETS AND FILTERS -->
  <LISTENER>
    <LISTENER-
CLASS>ORG.SPRINGFRAMEWORK.WEB.CONTEXT.CONTEXTLOADERLISTENER</LISTENER-CLASS>
  </LISTENER>
  <!-- PROVIDES ACCESS TO SESSION OUTSIDE OF DISPATCHERSERVLET -->
  <LISTENER>
    <LISTENER-
CLASS>ORG.SPRINGFRAMEWORK.WEB.CONTEXT.REQUEST.REQUESTCONTEXTLISTENER</LISTENER-
CLASS>
  </LISTENER>
  <!-- PROCESSES APPLICATION REQUESTS -->
  <SERVLET>
    <SERVLET-NAME>ACM</SERVLET-NAME>
    <SERVLET-
CLASS>ORG.SPRINGFRAMEWORK.WEB.SERVLET.DISPATCHERSERVLET</SERVLET-CLASS>
    <INIT-PARAM>
      <PARAM-NAME>CONTEXTCONFIGLOCATION</PARAM-NAME>
      <PARAM-VALUE>CLASSPATH:/SPRING/SERVLET-CONTEXT.XML</PARAM-
VALUE>
    </INIT-PARAM>
    <LOAD-ON-STARTUP>1</LOAD-ON-STARTUP>
  </SERVLET>
  <SERVLET-MAPPING>
    <SERVLET-NAME>ACM</SERVLET-NAME>
    <URL-PATTERN>/</URL-PATTERN>
  </SERVLET-MAPPING>
  <!-- SPRING SECURITY -->
  <FILTER>
    <FILTER-NAME>MULTIPARTFILTER</FILTER-NAME>
    <FILTER-
CLASS>ORG.SPRINGFRAMEWORK.WEB.MULTIPART.SUPPORT.MULTIPARTFILTER</FILTER-CLASS>
```

```

</FILTER>
<FILTER>
  <FILTER-NAME>SPRINGSECURITYFILTERCHAIN</FILTER-NAME>
  <FILTER-
CLASS>ORG.SPRINGFRAMEWORK.WEB.FILTER.DELEGATINGFILTERPROXY</FILTER-CLASS>
</FILTER>
<FILTER-MAPPING>
  <FILTER-NAME>MULTIPARTFILTER</FILTER-NAME>
  <URL-PATTERN>/*</URL-PATTERN>
</FILTER-MAPPING>
<FILTER-MAPPING>
  <FILTER-NAME>SPRINGSECURITYFILTERCHAIN</FILTER-NAME>
  <URL-PATTERN>/*</URL-PATTERN>
</FILTER-MAPPING>
<!-- ENFORCE UTF-8 ENCODING -->
<FILTER>
  <FILTER-NAME>SETCHARACTERENCODINGFILTER</FILTER-NAME>
  <FILTER-
CLASS>ORG.SPRINGFRAMEWORK.WEB.FILTER.CHARACTERENCODINGFILTER</FILTER-CLASS>
  <INIT-PARAM>
    <PARAM-NAME>ENCODING</PARAM-NAME>
    <PARAM-VALUE>UTF8</PARAM-VALUE>
  </INIT-PARAM>
  <INIT-PARAM>
    <PARAM-NAME>FORCEENCODING</PARAM-NAME>
    <PARAM-VALUE>TRUE</PARAM-VALUE>
  </INIT-PARAM>
</FILTER>
<FILTER-MAPPING>
  <FILTER-NAME>SETCHARACTERENCODINGFILTER</FILTER-NAME>
  <URL-PATTERN>/*</URL-PATTERN>
</FILTER-MAPPING>
<!-- DISABLE URL SESSION REWRITING AND SET SESSION TIMEOUT -->
<SESSION-CONFIG>
  <SESSION-TIMEOUT>30</SESSION-TIMEOUT>
  <TRACKING-MODE>COOKIE</TRACKING-MODE>
</SESSION-CONFIG>
</WEB-APP>

```

Table 5 Deployment descriptor for ACM prototype

The deployment descriptor of the ACM prototype is shown above. The Spring IoC container shared by all application servlets and filters is defined by referencing two Spring configuration files for the root context and the security context. A RequestContextListener is added to allow access to the session object from outside of the DispatcherServlet, which itself is defined by referencing a third Spring configuration file for the servlet context. This is followed by the configuration of Spring Security filters, ensuring that all requests to the application pass through the security layer. UTF8 encoding is enforced throughout the entire application, to avoid issues related to the charactersets used in application data. Finally cookie-based session tracking is enabled to avoid URL rewriting, which is identified as a security risk by Open Web App Security Project (OWASP). [59]

5.1.3.1 Spring Context

Spring Context provides an Inversion of Control (IoC) container, which is responsible for instantiating, configuring and assembling beans (i.e. instances of classes), by interpreting configuration metadata supplied as XML, annotation or Java code. Beans apply the Singleton software design pattern [60], i.e. there exists only a single bean per class, thus enabling beans to be referenced by class in other beans.

Spring Context depends on various other Spring modules such as: Spring Core, which provides a configuration model and dependency injection; Aspect-oriented Programming (AOP), an extension of Object-oriented Programming (OOP) which enables the modularization of horizontal concerns that cut across multiple types and objects [61]; and the Spring Expression Language (SpEL), an expression language with a syntax similar to EL that allows for method invocation and basic string templating [62].

As shown in Table 5 Deployment descriptor for ACM prototype, the configuration metadata for the IoC container is split over two files. The root context contains infrastructure beans such as data repositories and business services, and is shown below. The security context contains the application-specific settings of the security layer, which will be discussed in detail further below.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
  XMLNS:CONTEXT="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/CONTEXT"
  XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSchema-INSTANCE"
  XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD
  HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/CONTEXT
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/CONTEXT/SPRING-CONTEXT.XSD">
  <!-- ROOT CONTEXT: DEFINES SHARED RESOURCES VISIBLE TO ALL OTHER WEB
COMPONENTS -->
  <!-- SERVICE LAYER -->
  <BEAN ID="ACMFACADE" CLASS="COM.GKOTSCHY.ACM.FACADE.ACMFACADEACTIVITI" />
  <CONTEXT:COMPONENT-SCAN BASE-PACKAGE="COM.GKOTSCHY.ACM.FLOW" />
  <CONTEXT:COMPONENT-SCAN BASE-PACKAGE="COM.GKOTSCHY.ACM.VALIDATOR" />
  <CONTEXT:COMPONENT-SCAN BASE-PACKAGE="COM.GKOTSCHY.ACM.BPMN.BAS" />
  <!-- PROCESS LAYER -->
  <CONTEXT:COMPONENT-SCAN BASE-PACKAGE="COM.GKOTSCHY.ACM.BPMN.PS" />
  <!-- PERSISTENCE LAYER -->
  <CONTEXT:COMPONENT-SCAN BASE-PACKAGE="COM.GKOTSCHY.ACM.DAO" />
  <!-- LISTENERS -->
  <BEAN CLASS="COM.GKOTSCHY.ACM.SERVLET.ACMAUTHENTICATIONLISTENER" />
  <!-- ACCESSORS -->
  <BEAN CLASS="COM.GKOTSCHY.ACM.SPRING.APPLICATIONACCESSOR" />
  <!-- MESSAGE SOURCE -->
  <BEAN ID="MESSAGESOURCE"
CLASS="ORG.SPRINGFRAMEWORK.CONTEXT.SUPPORT.RELOADABLERESOURCEBUNDLEMESSAGESOURCE">
    <PROPERTY NAME="BASENAME" VALUE="/WEB-INF/MESSAGES"/>
    <PROPERTY NAME="DEFAULTENCODING" VALUE="UTF-8" />
    <PROPERTY NAME="CACHESECONDS" VALUE="60" />
  </BEAN>
  ...
</BEANS>
```

Table 6 Excerpt from root context of ACM prototype

As seen in the root context excerpt above, the persistence layer is defined by referencing the base package for all Data Access Objects (DAO), which are used to access the data entities stored directly by the application (i.e. the application's own DAOs do not offer access to entities stored by stand-alone components like Activiti or LDAP). The service layer is specified next and includes: the process execution façade of the ACM prototype; short-lived processes executed via Webflow; form validation logic for MVC-, Webflow- and Activiti-based forms; and business activity services encapsulating general business logic used by any or all long-lived processes executed via Activiti. The process layer contains process services, which (unlike business activity services) encapsulate specific business logic used by a certain long-lived process executed via Activiti. An authentication listener is defined to copy the authentication information for the acting user from the security context to the HTTP session, and an accessor enables the programmatic retrieval of beans by class or name. Finally a message source centralizes the management of static text, while providing support for parameterisation and internationalisation.

5.1.3.2 Spring MVC

Spring MVC provides hooks for the extension and customisation of web applications and RESTful web services. It is based on a Model-View-Controller architecture separating the responsibilities of encapsulating application data, rendering model data, and processing user requests. The central component is the DispatcherServlet, which handles all HTTP requests and responses. [63]

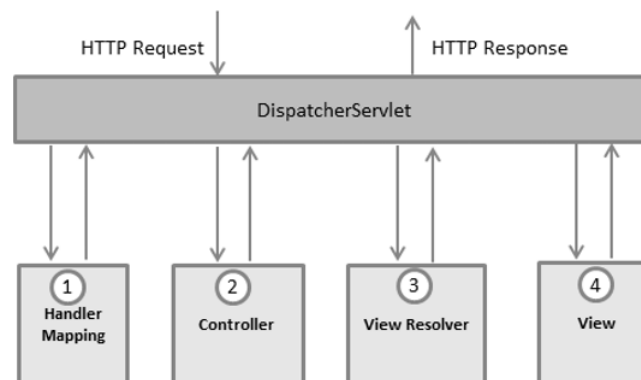


Figure 16 Spring MVC DispatcherServlet [64]

When an HTTP request is received, the DispatcherServlet consults the HandlerMapping to call the appropriate Controller, which routes the request to an appropriate service method, sets model data based on the implemented business logic, and finally identifies and returns the intended View for displaying the generated model. The DispatcherServlet then consults the ViewResolver to determine the correct View, and passes the model to the View, which finally renders it to the web browser. [64]

As shown in Table 5 Deployment descriptor for ACM prototype, the dispatcher servlet is configured via the servlet context, and contains controllers, view-resolvers and other web-related beans. If no appropriate bean is found in the servlet context, the request is delegated to the root context.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
```

```

<BEANS:BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC"
  XMLNS:BEANS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
  XMLNS:CONTEXT="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/CONTEXT"
  XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
  XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC/SPRING-MVC.XSD
  HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD
  HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/CONTEXT
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/CONTEXT/SPRING-CONTEXT.XSD">
  <!-- DISPATCHERSERVLET CONTEXT: DEFINES THIS SERVLET'S REQUEST-PROCESSING
INFRASTRUCTURE -->
  <!-- ENABLES THE SPRING MVC @CONTROLLER PROGRAMMING MODEL -->
  <ANNOTATION-DRIVEN />
  <!-- ENABLES ANNOTATION-BASED CONFIGURATION -->
  <CONTEXT:ANNOTATION-CONFIG />
  <!-- HANDLES HTTP GET REQUESTS FOR /RESOURCES/** BY EFFICIENTLY SERVING UP
STATIC RESOURCES IN THE ${WEBAPPROOT} DIRECTORY -->
  <RESOURCES MAPPING="/RESOURCES/**" LOCATION="/RESOURCES/" />
  <RESOURCES MAPPING="/CSS/**" LOCATION="/CSS/" />
  <RESOURCES MAPPING="/JS/**" LOCATION="/JS/" />
  <RESOURCES MAPPING="/FONTS/**" LOCATION="/FONTS/" />
  <RESOURCES MAPPING="/IMG/**" LOCATION="/IMG/" />
  <RESOURCES MAPPING="/FAVICON/**" LOCATION="/FAVICON/" />
  <RESOURCES MAPPING="/FAVICON.ICO" LOCATION="/" />
  <!-- PRESENTATION LAYER -->
  <CONTEXT:COMPONENT-SCAN BASE-PACKAGE="COM.GKOTSCHY.ACM.CONTROLLER" />
  <!-- LISTENERS -->
  <BEANS:BEAN CLASS="COM.GKOTSCHY.ACM.SERVLET.ACMHTTPSESSIONLISTENER" />
  <!-- INTERCEPTORS -->
  <INTERCEPTORS>
    <BEANS:BEAN
CLASS="COM.GKOTSCHY.ACM.SERVLET.ACMMVCHANDLERINTERCEPTOR" />
  </INTERCEPTORS>
  ...
</BEANS:BEANS>

```

Table 7 Excerpt from servlet context of ACM prototype

As seen in the servlet context excerpt above, annotation-based configuration is enabled for the beans managed by the DispatcherServlet. Requests for static resources (e.g. images, CSS and JavaScript code) are set up to bypass the MVC resolution process described in Figure 16 Spring MVC DispatcherServlet. The presentation layer is initialised by referencing the base package containing all Controllers. A SessionListener is set up to add listeners to the servlet by code, which makes it possible to inject Spring beans directly into the listener. Finally an MVCHandlerInterceptor is defined for recording the execution start time, current server time, and execution duration to the request context, as well as setting the authenticated user for requests directed at the Activiti process execution engine.

5.1.3.3 Spring Webflow

Spring Webflow is an extension of Spring MVC which allows executing short-lived, single-user workflows in a web application. A flow encapsulates a sequence of steps that guide a

single user through the execution of some business task. [65] Unlike business processes, flows are intended to be short-lived, and the flow state stored on the server-side is deleted after a set amount of time, if the flow is not completed before then. Also unlike business processes, there is no concept of swimlanes, as flows only concern the acting user, and do not involve other humans.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<BEANS:BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC"
  XMLNS:BEANS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
  XMLNS:WEBFLOW="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/WEBFLOW-CONFIG"
  XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
  XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC/SPRING-MVC.XSD
  HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD
  HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/WEBFLOW-CONFIG
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/WEBFLOW-CONFIG/SPRING-WEBFLOW-CONFIG.XSD">
...
  <!-- SPRING WEBFLOW -->
  <BEANS:BEAN ID="MVCVIEWFACTORYCREATOR"
CLASS="ORG.SPRINGFRAMEWORK.WEBFLOW.MVC.BUILDER.MVCVIEWFACTORYCREATOR">
    <BEANS:PROPERTY NAME="VIEWRESOLVERS" REF="TILESVIEWRESOLVER" />
  </BEANS:BEAN>
  <WEBFLOW:FLOW-BUILDER-SERVICES ID="FLOWBUILDERSERVICES" VIEW-FACTORY-
CREATOR="MVCVIEWFACTORYCREATOR" />
  <WEBFLOW:FLOW-REGISTRY ID="FLOWREGISTRY" BASE-PATH="/WEB-INF/FLOWS/" FLOW-
BUILDER-SERVICES="FLOWBUILDERSERVICES">
    <WEBFLOW:FLOW-LOCATION PATH="CREATE-PROCESS-INSTANCE/CREATE-PROCESS-
INSTANCE-FLOW.XML" ID="INSTANCES/CREATE/FLOW" />
    <WEBFLOW:FLOW-LOCATION PATH="CREATE-ATTACHMENT/CREATE-ATTACHMENT-
FLOW.XML" ID="ATTACHMENTS/CREATE/FLOW" />
  </WEBFLOW:FLOW-REGISTRY>
  <BEANS:BEAN
CLASS="ORG.SPRINGFRAMEWORK.WEBFLOW.MVC.SERVLET.FLOWHANDLERMAPPING">
    <BEANS:PROPERTY NAME="FLOWREGISTRY" REF="FLOWREGISTRY" />
    <BEANS:PROPERTY NAME="ORDER" VALUE="0" />
  </BEANS:BEAN>
  <WEBFLOW:FLOW-EXECUTOR ID="FLOWEXECUTOR" />
  <BEANS:BEAN CLASS="COM.GKOTSCHY.ACM.SERVLET.ACMFLOWHANDLERADAPTER">
    <BEANS:PROPERTY NAME="FLOWEXECUTOR" REF="FLOWEXECUTOR" />
  </BEANS:BEAN>
...
</BEANS:BEANS>
```

Table 8 Excerpt from servlet context of ACM prototype

As seen in the servlet context excerpt above, a ViewResolver is created and added to the FlowBuilderServices to enable Webflow to use Tiles for assembling page fragments. Two sample flows (managing the creation of process instances and attachments) are then registered and added to the FlowHandlerMapping. Finally a custom FlowHandlerAdapter is created to mimic the behaviour of the MVCHandlerInterceptor (shown in Table 7 Excerpt from servlet context of ACM prototype) in Webflow.

5.2 Application Architecture

The application architecture of the ACM prototype is composed of 5 layers: the security layer controls access to the application and its resources; the persistence layer store's the data structures used by the application; the service layer contains the implemented business logic; the process layer contains the customised business process execution engine; and the presentation layer consists of various components used to display the information generated by the service layer in a web browser.

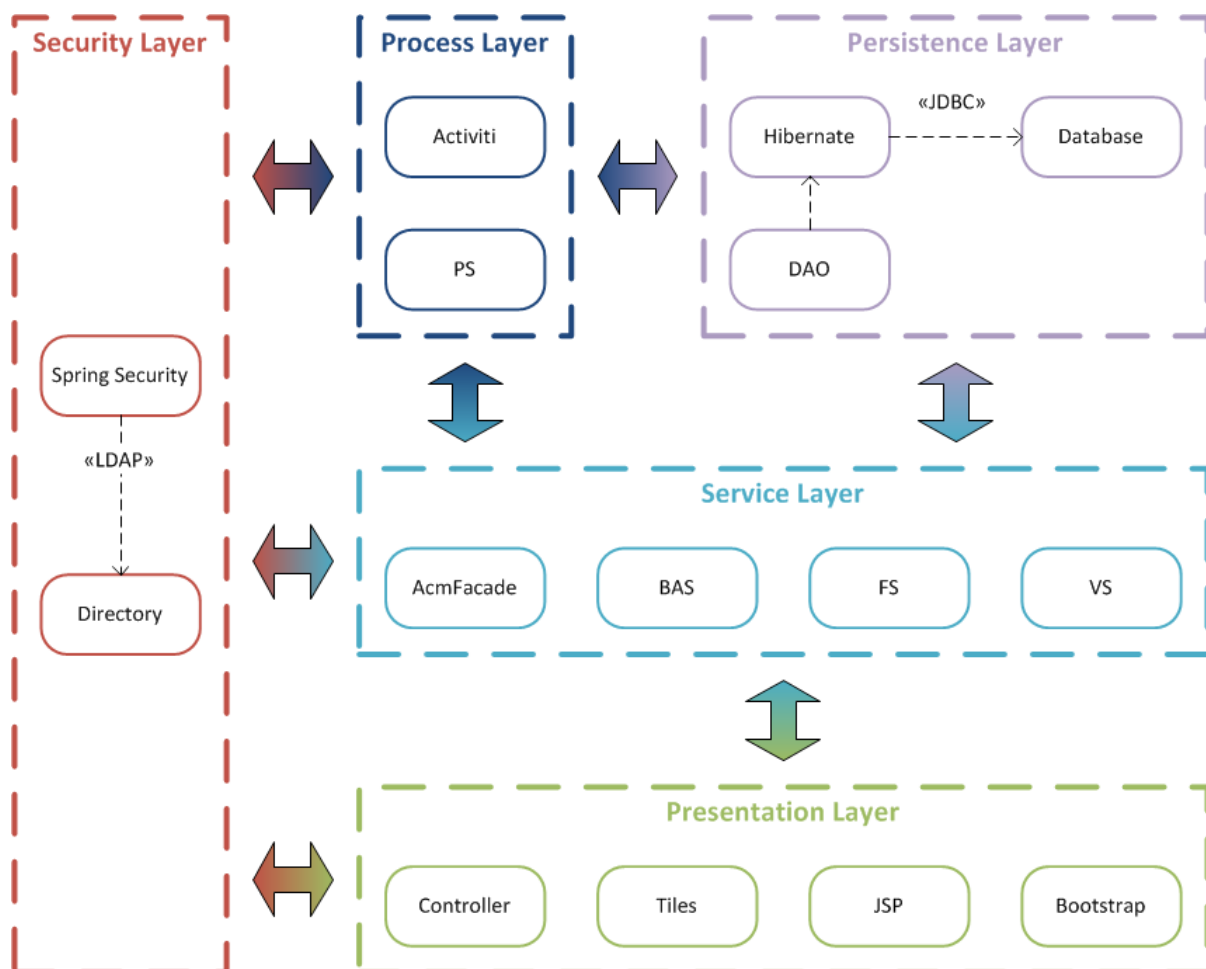


Figure 17 Application architecture of ACM prototype

5.2.1 Security Layer

The security layer of the ACM prototype prevents unauthenticated or unauthorised access, and depends on two components. Spring Security handles authentication and authorisation logic, while an LDAP directory server acts as both authentication provider and repository for users, groups and roles. Functional access is defined in the application configuration, and resource-based access is granted based on an Access Control List (ACL) managed by Activiti.

5.2.1.1 Spring Security

Spring Security is an authorisation and access control framework, which is the de facto standard solution for securing Spring web applications. When a client sends a request, the container in which an application is deployed decides which filters and servlet apply, based on the path specified in the request. A filter can modify the request or response, and block the further processing of a request, thus denying access to the application. Spring Security is installed as a single filter in the chain between client and servlet, but represents multiple filter chains itself. [66]

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<BEANS:BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/SECURITY"
  XMLNS:BEANS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
  XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
  XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/SECURITY
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/SECURITY/SPRING-SECURITY.XSD
  HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD">
  <!-- LEAVE STATIC CSS, JAVASCRIPT, FONTS AND IMAGES UNPROTECTED -->
  <HTTP PATTERN="/CSS/**" SECURITY="NONE"/>
  <HTTP PATTERN="/JS/**" SECURITY="NONE"/>
  <HTTP PATTERN="/FONTS/**" SECURITY="NONE"/>
  <HTTP PATTERN="/IMG/**" SECURITY="NONE"/>
  <HTTP PATTERN="/FAVICON/**" SECURITY="NONE"/>
  <HTTP PATTERN="/FAVICON.ICO" SECURITY="NONE"/>
  <HTTP USE-EXPRESSIONS="TRUE">
    <INTERCEPT-URL PATTERN="/LOGIN" ACCESS="ISANONYMOUS()"/>
    <INTERCEPT-URL PATTERN="/LOGOUT" ACCESS="PERMITALL"/>
    <INTERCEPT-URL PATTERN="/PING" ACCESS="PERMITALL"/>
    <INTERCEPT-URL PATTERN="/*" ACCESS="ISAUTHENTICATED()" />
    <FORM-LOGIN LOGIN-PAGE="/LOGIN" AUTHENTICATION-FAILURE-
URL="/LOGIN?FAILED" LOGIN-PROCESSING-URL="/AUTHENTICATE" USERNAME-
PARAMETER="USERNAME" PASSWORD-PARAMETER="PASSWORD" />
    <LOGOUT LOGOUT-URL="/LOGOUT" />
    <REMEMBER-ME KEY="ACM" />
  </HTTP>
  <!-- FILE UPLOAD -->
  <BEANS:BEAN ID="FILTERMULTIPARTRESOLVER"
CLASS="ORG.SPRINGFRAMEWORK.WEB.MULTIPART.COMMONS.COMMONSMULTIPARTRESOLVER">
    <BEANS:PROPERTY NAME="MAXUPLOADSIZE" VALUE="1000000"/>
  </BEANS:BEAN>
  <!-- IN-MEMORY LDAP SERVER AT LOCALHOST:33389 USERNAME:UID=ADMIN,OU=SYSTEM
PASSWORD:SECRET -->
  <LDAP-SERVER ID="LDAPSERVER" ROOT="DC=GKOTSCHY,DC=COM"
LDIF="CLASSPATH:/LDAP/ACM.LDIF" MANAGER-DN="UID=ADMIN,OU=SYSTEM" MANAGER-
PASSWORD="SECRET" />
  <AUTHENTICATION-MANAGER>
    <!-- LDAP AUTHENTICATION PROVIDER -->
    <LDAP-AUTHENTICATION-PROVIDER SERVER-REF="LDAPSERVER" USER-SEARCH-
FILTER="(UID={0})" USER-SEARCH-BASE="OU=PEOPLE" GROUP-SEARCH-
BASE="OU=ROLES,OU=ACM,OU=GROUPS" USER-DETAILS-CLASS="INETORGPPERSON" ROLE-
PREFIX="" />
  </AUTHENTICATION-MANAGER>
```

```

<!-- REMEMBER-ME AUTHENTICATION NEEDS USERSERVICE BUT DUPLICATES LDAP-
AUTHENTICATION-PROVIDER CONFIG -->
<LDAP-USER-SERVICE ID="LDAPUSERSERVICE" SERVER-REF="LDAPSERVER" USER-
SEARCH-FILTER="(UID={0})" USER-SEARCH-BASE="OU=PEOPLE" GROUP-SEARCH-
BASE="OU=ROLES,OU=ACM,OU=GROUPS" USER-DETAILS-CLASS="INETORGPERSO" ROLE-
PREFIX="" />
</BEANS:BEANS>

```

Table 9 Excerpt from security context of ACM prototype

As seen in the security context excerpt above, several patterns are specified to exempt certain static content (e.g. images, CSS and JavaScript code) from access control. Next a list of patterns is configured that allow access to anonymous (e.g. login page), all (e.g. logout page) or authenticated (e.g. any page not specified by a previous filter) users. The login functionality is configured to accept username and password for authentication, and remember-me functionality is enabled to allow users to stay authenticated via a locally-stored cookie. A resolver is defined, to handle file uploads, followed by an in-memory LDAP server. The in-memory version of ApacheDS used by the ACM prototype is deprecated and should only be used for development purposes. A production system would instead access an external LDAP server, e.g. the stand-alone version of ApacheDS. [67] An authentication provider is set up referencing the in-memory LDAP configured above, and finally an LDAP user service is explicitly defined. This definition is required by the Spring Security remember-me authentication, and causes an unfortunate duplication of the LDAP authentication provider configuration.

5.2.1.2 LDAP

The Lightweight Directory Access Protocol (LDAP) is an open industry standard for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. [68] Using a standard protocol prevents the ACM prototype's dependency on a specific authentication provider implementation, and allows for accessing and maintaining information on users, groups and roles. Activiti offers integration with the LDAP protocol, allowing access to this information via the Activiti API.

```

<?XML VERSION="1.0" ENCODING="UTF-8"?>
<BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD">
...
<BEAN ID="LDAPCONFIGURATOR"
CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMLDAPCONFIGURATOR">
<PROPERTY NAME="SERVER" VALUE="LDAP://LOCALHOST" />
<PROPERTY NAME="PORT" VALUE="33389" />
<PROPERTY NAME="USER" VALUE="UID=ADMIN,OU=SYSTEM" />
<PROPERTY NAME="PASSWORD" VALUE="SECRET" />
<PROPERTY NAME="USERBASEDN" VALUE="OU=PEOPLE,DC=GKOTSCHY,DC=COM" />
<PROPERTY NAME="GROUPBASEDN"
VALUE="OU=ACM,OU=GROUPS,DC=GKOTSCHY,DC=COM" />
<PROPERTY NAME="QUERYUSERBYUSERID"
VALUE="(&((OBJECTCLASS=INETORGPERSO)(UID={0}))" />
<PROPERTY NAME="QUERYUSERBYFULLNAMELIKE"
VALUE="(&((OBJECTCLASS=INETORGPERSO)(|({0}={1}*)({2}={3}*))" />

```

```

    <PROPERTY NAME="QUERYGROUPSFORUSER"
VALUE="(&AMP;(OBJECTCLASS=GROUPOFUNIQUE NAMES)(UNIQUEMEMBER={0}))" />
    <PROPERTY NAME="USERIDATTRIBUTE" VALUE="UID" />
    <PROPERTY NAME="USERFIRSTNAMEATTRIBUTE" VALUE="GN" />
    <PROPERTY NAME="USERLASTNAMEATTRIBUTE" VALUE="SN" />
    <PROPERTY NAME="USEREMAILATTRIBUTE" VALUE="MAIL" />
    <PROPERTY NAME="USERNAMEATTRIBUTE" VALUE="DISPLAYNAME" />
    <PROPERTY NAME="USERNAMESATTRIBUTE" VALUE="CN" />
    <PROPERTY NAME="USERTENANTATTRIBUTE" VALUE="O" />
    <PROPERTY NAME="USERCARLICENSEATTRIBUTE" VALUE="CARLICENSE" />
    <PROPERTY NAME="USERDEPARTMENTNUMBERATTRIBUTE"
VALUE="DEPARTMENTNUMBER" />
    <PROPERTY NAME="USERDESCRIPTIONATTRIBUTE" VALUE="DESCRIPTION" />
    <PROPERTY NAME="USEREMPLOYEEENUMBERATTRIBUTE" VALUE="EMPLOYEEENUMBER"
/>
    <PROPERTY NAME="USERHOMEPHONEATTRIBUTE" VALUE="HOMEPHONE" />
    <PROPERTY NAME="USERHOMEPOSTALADDRESSATTRIBUTE"
VALUE="HOMEPOSTALADDRESS" />
    <PROPERTY NAME="USERINITIALSATTRIBUTE" VALUE="INITIALS" />
    <PROPERTY NAME="USERMOBILEATTRIBUTE" VALUE="MOBILE" />
    <PROPERTY NAME="USERPOSTALADDRESSATTRIBUTE" VALUE="POSTALADDRESS" />
    <PROPERTY NAME="USERPOSTALCODEATTRIBUTE" VALUE="POSTALCODE" />
    <PROPERTY NAME="USERSTREETATTRIBUTE" VALUE="STREET" />
    <PROPERTY NAME="USERTELEPHONENUMBERATTRIBUTE"
VALUE="TELEPHONENUMBER" />
    <PROPERTY NAME="USERTITLEATTRIBUTE" VALUE="TITLE" />
    <PROPERTY NAME="GROUPIDATTRIBUTE" VALUE="CN" />
    <PROPERTY NAME="GROUPNAMEATTRIBUTE" VALUE="DISPLAYNAME" />
    <PROPERTY NAME="GROUPTYPEATTRIBUTE" VALUE="OBJECTCLASS" />
    <PROPERTY NAME="GROUPDNATTRIBUTE" VALUE="DN" />
    <PROPERTY NAME="GROUPOUATTRIBUTE" VALUE="OU" />
    <PROPERTY NAME="GROUPMEMBERATTRIBUTE" VALUE="UNIQUEMEMBER" />
    <PROPERTY NAME="LDAPQUERYBUILDER">
        <BEAN CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMLDAPQUERYBUILDER">
            <PROPERTY NAME="QUERYORGANISATIONS"
VALUE="(OBJECTCLASS=ORGANIZATIONALUNIT)" />
            <PROPERTY NAME="QUERYORGANISATIONSBYID"
VALUE="(&AMP;(OBJECTCLASS=ORGANIZATIONALUNIT)(OU={0}*)" />
            <PROPERTY NAME="QUERYROLES"
VALUE="(OBJECTCLASS=GROUPOFUNIQUE NAMES)" />
            <PROPERTY NAME="QUERYROLESBYID"
VALUE="(&AMP;(OBJECTCLASS=GROUPOFUNIQUE NAMES)(CN={0}*)" />
        </BEAN>
    </PROPERTY>
    <PROPERTY NAME="LDAPGROUPMANAGERFACTORY">
        <BEAN
CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMLDAPGROUPMANAGERFACTORY">
            <CONSTRUCTOR-ARG NAME="LDAPCONFIGURATOR"
REF="LDAPCONFIGURATOR" />
            <CONSTRUCTOR-ARG NAME="CLOCKREADER" REF="ACTIVITICLOCK"
/>
        </BEAN>

```

```

        </PROPERTY>
        <PROPERTY NAME="LDAPUSERMANAGERFACTORY">
            <BEAN
CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMLDAPUSERMANAGERFACTORY">
                <CONSTRUCTOR-ARG NAME="LDAPCONFIGURATOR"
REF="LDAPCONFIGURATOR" />
            </BEAN>
        </PROPERTY>
    </BEAN>
    ...
</BEANS>

```

Table 10 Excerpt from root context of ACM prototype

The root context excerpt above shows how Activiti is configured to use an LDAP directory server as user and group repository. The basic settings mirror the values defined in the security context, and are followed by a series of standard query definitions (e.g. QueryGroupsForUser which searches the LDAP directory for all GroupOfUniqueNames that contain a member matching a specific username). Next the user and group attributes stored in the LDAP directory are registered, allowing a custom LDAPUserManager to map search results to the data structures used by the ACM prototype. An LDAPQueryBuilder allows for the creation of custom queries (e.g. QueryRoles which returns a list of the roles stored in the LDAP directory) in code. Finally a custom UserManager and GroupManager are defined using the factory method pattern.

```

PUBLIC CLASS ACMLDAPGROUPMANAGER EXTENDS ABSTRACTMANAGER IMPLEMENTS
GROUPIDENTITYMANAGER {
    ...
    PUBLIC LIST<GROUP> FINDGROUPSBYUSER(FINAL STRING USERID) {
        IF (LDAPGROUPCACHE != NULL) {
            // FIRST TRY THE CACHE (IF ONE IS DEFINED)
            LIST<GROUP> GROUPS = LDAPGROUPCACHE.GET(USERID);
            IF (GROUPS != NULL) {
                RETURN GROUPS;
            }
        }
        LDAPTEMPLATE LDAPTEMPLATE = NEW LDAPTEMPLATE(LDAPCONFIGURATOR);
        RETURN LDAPTEMPLATE.EXECUTE(NEW LDAPCALLBACK<LIST<GROUP>>()) {
            PUBLIC LIST<GROUP> EXECUTEINCONTEXT(INITIALDIRCONTEXT
INITIALDIRCONTEXT) {
                // DO THE SEARCH AGAINST LDAP
                LIST<GROUP> GROUPS = NEW ARRAYLIST<GROUP>();
                TRY {
                    STRING BASEDN = LDAPCONFIGURATOR.GETGROUPBASEDN()
!= NULL ? LDAPCONFIGURATOR.GETGROUPBASEDN() : LDAPCONFIGURATOR.GETBASEDN();
                    STRING SEARCHEXPRESSION =
LDAPCONFIGURATOR.GETLDAPQUERYBUILDER().BUILDQUERYGROUPSFORUSER(LDAPCONFIGURATOR,
USERID);
                    NAMINGENUMERATION<?> NAMINGENUM =
INITIALDIRCONTEXT.SEARCH(BASEDN, SEARCHEXPRESSION,
CREATESEARCHCONTROLS(SEARCHCONTROLS.SUBTREE_SCOPE));
                    WHILE (NAMINGENUM.HASMORE()) {

```

```

NAMINGENUM.NEXT();
                                ACMROLEENTITY GROUP = NEW ACMROLEENTITY();
                                SEARCHRESULT RESULT = (SEARCHRESULT)
                                MAPSEARCHRESULTTOROLE(RESULT, GROUP);
                                GROUPS.ADD(GROUP);
                                }
                                NAMINGENUM.CLOSE();
                                IF (LDAPGROUPCACHE != NULL) {
                                    // CACHE RESULTS FOR LATER
                                    LDAPGROUPCACHE.ADD(USERID, GROUPS);
                                }
                                RETURN GROUPS;
                                } CATCH (NAMINGEXCEPTION E) {
                                    THROW NEW ACTIVITIEXCEPTION("COULD NOT FIND
GROUPS FOR USER " + USERID, E);
                                }
                            });
                        }
                    ...
                }
            }

```

Table 11 Excerpt from AcmlLDAPGroupManager source code

In the sample code shown above, we can see how the previously discussed QueryGroupsForUser is executed via the Activiti API. First the cache is checked in case the same query has already been executed recently. Applying the Command software design pattern [60], an LdapTemplate is then used to execute query logic defined as an inline object. First the appropriate node of the LDAP directory is identified, then a search expression is built using the supplied username, and the search result is mapped to a data structure. Finally the search result is added to the cache to optimise subsequent queries.

5.2.2 Persistence Layer

The main responsibility of the persistence layer is the storage and retrieval of data. Data Access Objects (DAO) encapsulate the details of an underlying persistence API like Java Database Connectivity (JDBC), by translating all native exceptions and data entities into structures understood by the application, and serve as the single point of access for all requests originating from the service layer. [69] The DAOs rely on Hibernate, which is an Object Relational Mapping framework integrated into the application via Spring ORM. [70] It allows for the definition of database structures and queries in code, and provides transaction management. Unfortunately using abstraction layers like Hibernate causes a loss of performance, and therefore the process layer uses native database queries, bypassing both the DAOs and Hibernate to access the database directly. The final abstraction layer is JDBC, a standard API that prevents the application from depending on a specific database implementation. [71] Similar to the effect that LDAP has in the security layer, this allows easily switching implementations well-suited for development (i.e. in-memory) with ones optimised for production (i.e. stand-alone).

```

<?XML VERSION="1.0" ENCODING="UTF-8"?>
<BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
XMLNS:TX="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/TX"

```

```

XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/TX
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/TX/SPRING-TX.XSD">
...
<!-- DATABASE -->
<BEAN ID="DATASOURCE" CLASS="ORG.APACHE.COMMONS.DBCP.BASICDATASOURCE"
DESTROY-METHOD="CLOSE">
  <PROPERTY NAME="DRIVERCLASSNAME" VALUE="ORG.HSQLDB.JDBCDBDRIVER"/>
  <PROPERTY NAME="URL" VALUE="JDBC:HSQldb:MEM:ACM"/>
  <PROPERTY NAME="USERNAME" VALUE="SA"/>
  <PROPERTY NAME="PASSWORD" VALUE="" />
</BEAN>
<BEAN ID="SESSIONFACTORY"
CLASS="ORG.SPRINGFRAMEWORK.ORM.HIBERNATE5.LOCALSESSIONFACTORYBEAN">
  <PROPERTY NAME="DATASOURCE" REF="DATASOURCE"/>
  <PROPERTY NAME="ANNOTATEDCLASSES">
    <LIST>
      <VALUE>COM.GKOTSCHY.ACM.ENTITY.PINGENTITY</VALUE>
    </LIST>
  </PROPERTY>
  <PROPERTY NAME="HIBERNATEPROPERTIES">
    <PROPS>
      <PROP
KEY="HIBERNATE.DIALECT">ORG.HIBERNATE.DIALECT.HSQLDIALECT</PROP>
      <PROP KEY="HIBERNATE.HBM2DDL.AUTO">CREATE</PROP>
      <PROP
KEY="HIBERNATE.HBM2DDL.IMPORT_FILES">/REFERENCE_DATA.SQL</PROP>
      <PROP
KEY="ORG.HIBERNATE.ENVERS.AUDIT_TABLE_SUFFIX">_AUDITLOG</PROP>
      <PROP
KEY="ORG.HIBERNATE.ENVERS.REVISION_FIELD_NAME">REVISIONID</PROP>
      <PROP
KEY="ORG.HIBERNATE.ENVERS.REVISION_TYPE_FIELD_NAME">REVISIONTYPE</PROP>
    </PROPS>
  </PROPERTY>
</BEAN>
<BEAN ID="TRANSACTIONMANAGER"
CLASS="ORG.SPRINGFRAMEWORK.ORM.HIBERNATE5.HIBERNATETRANSACTIONMANAGER">
  <PROPERTY NAME="SESSIONFACTORY" REF="SESSIONFACTORY" />
</BEAN>
<TX:ANNOTATION-DRIVEN TRANSACTION-MANAGER="TRANSACTIONMANAGER"/>
...
</BEANS>

```

Table 12 Excerpt from root context of ACM prototype

The root context excerpt above shows the configuration of the persistence layer in Spring. A data source is defined using an Apache Commons implementation, and references a database via JDBC. The ACM prototype uses a single database to store the business objects managed by the application, as well as the data structures managed by Activiti. An in-memory HyperSQL DataBase (HSQLDB) was chosen for ease-of-use during

development, while a production environment would likely feature a more performant database like Oracle DB, MySQL or PostgreSQL. Next a Hibernate SessionFactory is configured using the previously defined data source, and the business objects managed by Hibernate are registered. As the ACM prototype concerns itself mainly with process execution (and process execution entities are managed by Activiti), only a single sample entity is implemented. HSQL is specified as the native dialect of the data source, and (again to facilitate development) Hibernate is configured to create a clean schema every time the application is started. This is followed by a reference to an SQL file containing static reference data, which is loaded after the schema is created. The final three properties concern Envers, which is a Hibernate module that aims to provide an easy-to-use auditing and versioning layer for business objects. [72] Envers records all changes made to registered business objects by storing a revision entity containing the state of the business object before it was changed, and provides an API to access this historical data. While Envers covers the need for auditing and versioning of business objects managed by the application, Activiti provides its own historic record in the form of Activiti History, which captures and stores all events and state changes occurring during the execution of a process. Finally a transaction manager is created, which once again only works with the business objects managed by the application, while Activiti provides its own transaction model for process execution.

5.2.3 Service Layer

The service layer contains the business logic implemented by the ACM prototype, with the exception of business logic specific to a certain process model, which is contained in the process layer (see 5.2.4). The business logic is contained in beans which are deployed in (and primarily accessed through) the IoC container. Spring MVC further allows to easily expose beans via the Simple Object Access Protocol (SOAP) or Representational State Transfer (REST), thus promoting external re-use of business logic implemented by the application. The ACM prototype implements the Façade software design pattern [60], channelling all requests to the process execution through a single bean, the AcmFacade.

5.2.3.1 AcmFacade

The central role of the AcmFacade in the service landscape of the ACM prototype is shown in Figure 18 Role of AcmFacade in service landscape. Any request originating from the controller, or from a flow, validation or business activity service, is routed through the AcmFacade. The request is then either treated by the façade which has access to the persistence layer via DAOs, or it is relayed to an Activiti service in the process layer, which in turn accesses the persistence layer directly. By routing all requests through the façade, horizontal concerns like resource-based access control can be enforced on process entities, and the application itself can be decoupled from the implemented execution engine.

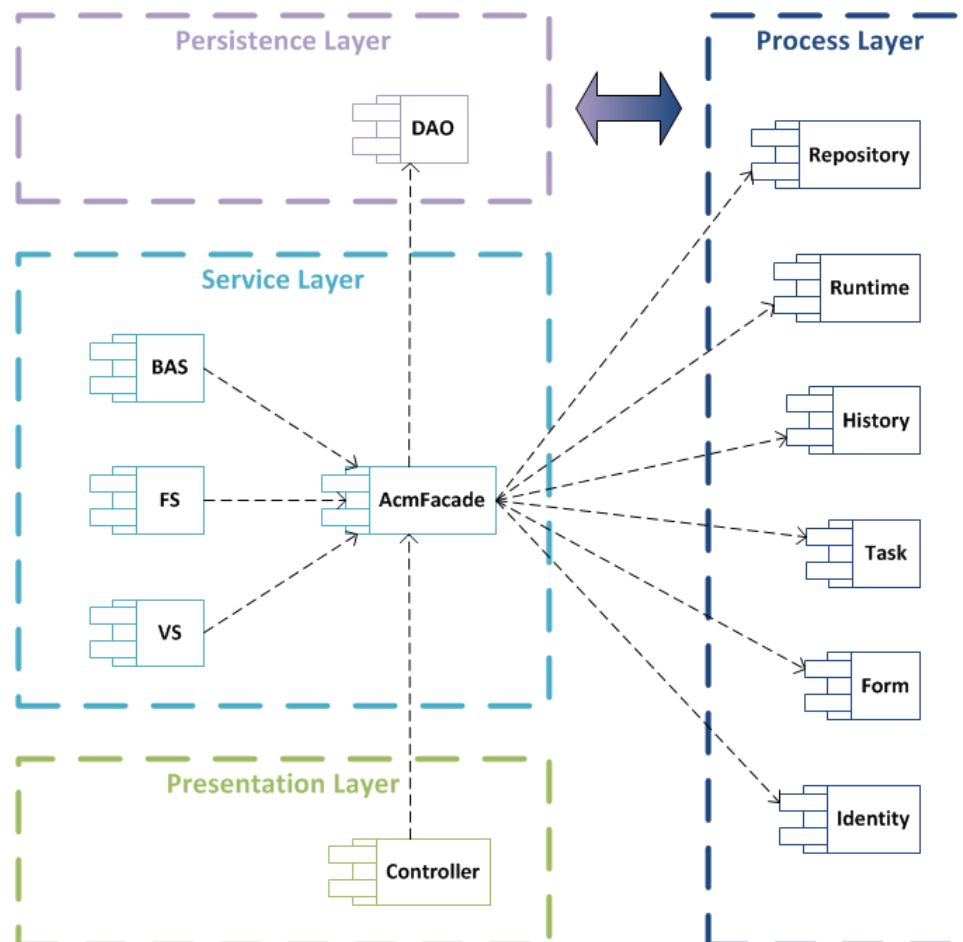


Figure 18 Role of AcmFacade in service landscape

Activiti services are stateless, allowing for cluster-based execution on multiple nodes, with each instance of Activiti connecting to the same data source. AcmFacade encapsulates the following Activiti services. [73]

- **Repository service:** The repository service is responsible for the management of process definitions and deployments. A deployment is a container for various process resources such as process definitions, Java-based code or graphical representations of a process. The service allows to deploy, query, suspend, activate, and retrieve these process resources.
- **Run-time service:** The run-time service allows for the creation of process instances from process definitions, and handles all interactions with a running process instance. This includes querying process instances, retrieving and storing process variables, and calling active process signals.
- **History service:** The history service offers access to the historical data collected by Activiti during process execution, e.g. start and completion times, acting users, and the sequence of activities (i.e. path) followed by the process instance.
- **Task service:** The task service handles all request related to user tasks. This includes querying, claiming and completing tasks, as well as managing task assignment and creating new tasks at run-time.

- **Form service:** The form service enables the retrieval and submission of forms defined directly in the process definition. Start forms are filled out by a user when creating a process instance, and the data they contain is injected into the process state before execution begins. Task forms are filled out by a user when completing a user task, and the data they contain is injected into the process state before execution resumes.
- **Identity service:** The identity service allows for the creation, querying, updating, or deletion of users and groups stored on a directory server.

While the AcmFacade depends on Activiti as process execution engine, a different implementation of the same interface could use a different process execution engine (if that engine provided at least the same capabilities as the customised version of Activiti contained in the ACM prototype). The implemented behaviour can be grouped into 9 categories.

- **Can:** Determines whether a given user can perform a certain action on a specific process model, definition, instance or task.
- **Do:** Performs an action on a certain process entity, e.g. claiming a task, deleting a comment, or suspending a process.
- **Find:** Uses the Activiti API to query or retrieve process entities, e.g. retrieving all comments associated with a certain process instance, querying for all process definitions associated to a certain model, or retrieving a certain user from the LDAP directory.
- **Get:** Exposes certain technical information, e.g. the access control list for a certain process entity, the available signals for a certain process instance or task, or the graphical representation of a process definition.
- **Is:** Checks the ACL of a certain process entity, to determine whether a given user has a specific association with that entity, thus allowing him to perform certain actions on that entity.
- **Load:** Loads certain metadata for process entities, e.g. the ACL for a process model, definition, instance or task.
- **Query:** Provides various customised queries, e.g. filtering process models and definitions by tenant, excluding subprocesses from process instance queries, or filtering tasks by user.
- **Resolve:** Resolves data entities retrieved from Activiti into Data Transfer Objects (DTO), which are used to display information in the presentation layer.
- **Validate:** Contains validation logic to determine whether the acting user has a certain username or belongs to a specific group or tenant.

5.2.3.2 Business Activity Services (BAS)

The ACM prototype relies on different types of service models (i.e. business activity, flow, validation and process) relating to specific functional contexts of encapsulated capabilities. While the AcmFacade provides a single point of access to the services of the process execution engine, BAS represent an atomic business logic that can be a part of one or several IT processes. The process layer is protected against changes in the underlying implementations of this atomic business logic, which, unlike the business logic contained in process services, does not directly depend on a specific process model.

A tangible example of a BAS is a notification service. At various points in a case, one or more stakeholders (e.g. Member States) have to be notified of certain events (e.g. publication of a customs decision). The atomic action of notifying someone of something can

be taken in various contexts, i.e. different process models or even outside of process execution. By implementing the atomic business logic of notification as a BAS, it is made available any potential consumers inside of the IT system, and can be exposed as a web service to consumers outside of the IT system. [74]

5.2.3.3 Flow Services (FS)

For the purposes of this thesis we want to concentrate on the execution of long-lived distributed workflows via Activiti, yet Spring Webflow offers a very interesting take on executing short-lived single-user workflows. The logic required by such flows is encapsulated in flow services, which are part of the service layer, and depend on the AcmFacade for accessing the process or persistence layers.

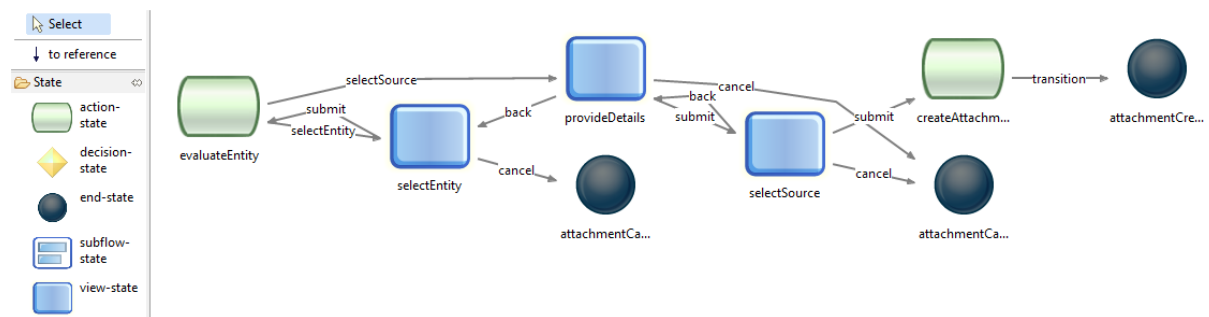


Figure 19 Create attachment webflow

The sample webflow shown above depicts the create attachment workflow, which associates unstructured information (e.g. documents or emails) with process entities. The webflow begins by evaluating the selected entity. If no valid entity has been provided, the selectEntity view is displayed, allowing the user to select a process instance or task, before the webflow loops back to the evaluateEntity action. Once the validity of the selected entity has been confirmed, the user can enter metadata in the provideDetails view, and then select the source (either as physical file or by URL reference) in the selectSource view. Once all the information required to create an attachment has been collected, the createAttachment action performs the necessary logic by calling the Activiti TaskService (which handles attachments for both process instances and tasks) through the AcmFacade.

Webflows are short-lived single-user workflows (see chapter 5.1.3.3), and Spring Webflow essentially acts as a server-side state engine. By storing the state of the entire workflow at each step, Spring Webflow allows the user to navigate backwards and forwards through a pre-defined sequence of actions, decisions and views. The information collected during the workflow from various forms filled out by the acting user, is stored on the server, and used to perform a single complex action, e.g. the creation of an attachment.

5.2.3.4 Validation Services (VS)

The ACM prototype allows for data entry through various channels, such as simple web interfaces (i.e. Spring MVC), short-lived process execution (i.e. Spring Webflow), and long-lived process execution (i.e. Activiti). VS provide a common approach to data validation across all channels, and rely on the AcmFacade to access existing data required for validation of form input. Unlike native Activiti forms, which only validate data type and whether an attribute is required, readable or writeable, VS allow forms to be checked through custom application logic, e.g. to determine whether an entity referenced by type and ID actually exists.

5.2.4 Process Layer

The process layer contains the components required to execute long-running and distributed workflows. This includes the process execution engine and its services (described in chapter 5.2.3.1), as well as the specific business logic required by the deployed process models.

5.2.4.1 Process Services (PS)

The process services model encapsulates a set of capabilities representing steps in a typically long-running end-to-end business process, and is a typical example of service composition. Each process service serves a specific process model, and contains business logic orchestrated by the process definitions contained in that model. Unlike BAS, this business logic is not meant to be re-usable by other process models or outside of process execution.

5.2.4.2 Activiti

Alfresco Activiti is a Business Process Management (BPM) solution targeted at business people and developers, based on a high performance business process execution engine promising the flexibility and scalability to handle a wide variety of processes. [75] It is open source, written in Java, and supports processes described using the Business Process Model and Notation (BPMN) version 2.0. The open source version of Activiti relies on an Eclipse plug-in to model custom elements, attributes and properties, while the enterprise version sold by Alfresco features a full BPMN editor. [76]

The strength of Activiti is its light-weight approach, which allows for both deep customisation and strong integration with the web application. Other open source process execution engines exist, e.g. jBPM and Camunda (which is itself an Activiti fork created in 2013 [77]), and two key engineers (Joram Barrez and Tijs Rademakers) left Alfresco in 2016 to create their own Activiti fork, aiming to support execution of both BPMN and CMMN. [78] The ACM prototype used in this thesis was developed in late 2014, and the choice of Activiti as execution engine was based on the post-mortem analysis of the NEST prototype, which used a similar open source-based technology stack.

```
<?XML VERSION="1.0" ENCODING="UTF-8"?>
<BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
      XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
      XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
      HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD">
  ...
  <!-- ACTIVITI -->
  <BEAN ID="RUNTIMESERVICE" FACTORY-BEAN="PROCESSENGINE" FACTORY-
  METHOD="GETRUNTIMESERVICE" />
  <BEAN ID="REPOSITORYSERVICE" FACTORY-BEAN="PROCESSENGINE" FACTORY-
  METHOD="GETREPOSITORYSERVICE" />
  <BEAN ID="TASKSERVICE" FACTORY-BEAN="PROCESSENGINE" FACTORY-
  METHOD="GETTASKSERVICE" />
  <BEAN ID="MANAGEMENTSERVICE" FACTORY-BEAN="PROCESSENGINE" FACTORY-
  METHOD="GETMANAGEMENTSERVICE" />
  <BEAN ID="IDENTITYSERVICE" FACTORY-BEAN="PROCESSENGINE" FACTORY-
  METHOD="GETIDENTITYSERVICE" />
  <BEAN ID="HISTORYSERVICE" FACTORY-BEAN="PROCESSENGINE" FACTORY-
  METHOD="GETHISTORYSERVICE" />
  <BEAN ID="FORMSERVICE" FACTORY-BEAN="PROCESSENGINE" FACTORY-
```

```

METHOD="GETFORMSERVICE" />
  <BEAN ID="ACTIVITICLOCK"
CLASS="ORG.ACTIVITI.ENGINE.IMPL.UTIL.DEFAULTCLOCKIMPL" />
  <BEAN ID="PROCESSENGINECONFIGURATION"
CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMPROCESSENGINECONFIGURATION">
  <PROPERTY NAME="DATABASETYPE" VALUE="H2" />
  <PROPERTY NAME="DATABASESCHEMAUPDATE" VALUE="TRUE" />
  <PROPERTY NAME="DATASOURCE" REF="DATASOURCE" />
  <PROPERTY NAME="TRANSACTIONMANAGER" REF="TRANSACTIONMANAGER" />
  <PROPERTY NAME="JOBEXECUTORACTIVATE" VALUE="FALSE" />
  <PROPERTY NAME="HISTORY" VALUE="FULL" />
  <PROPERTY NAME="DEPLOYMENTRESOURCES"
VALUE="CLASSPATH:/BPMN/EC/TRADE/FTA/FTA-V1.ZIP" />
  <PROPERTY NAME="DEPLOYMENTNAME" VALUE="FREE TRADE AGREEMENT" />
  <PROPERTY NAME="DEPLOYMENTCATEGORY" VALUE="FREE-TRADE-AGREEMENT" />
  <PROPERTY NAME="DEPLOYMENTTENANTID"
VALUE="ORGANISATIONS.EU.EC.TRADE" />
  <PROPERTY NAME="DEPLOYMENTSTARTER" VALUE="ACM" />
  <PROPERTY NAME="DEPLOYMENTOWNER"
VALUE="GROUP:ORGANISATIONS.EU.EC.TRADE.A.4" />
  <PROPERTY NAME="RUNTIMESERVICE">
    <BEAN CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMRUNTIMESERVICEIMPL"
/>
  </PROPERTY>
  <PROPERTY NAME="TASKSERVICE">
    <BEAN CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMTASKSERVICEIMPL" />
  </PROPERTY>
  <PROPERTY NAME="FORMSERVICE">
    <BEAN CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMFORMSERVICEIMPL" />
  </PROPERTY>
  <PROPERTY NAME="IDENTITYSERVICE">
    <BEAN CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMIDENTITYSERVICEIMPL"
/>
  </PROPERTY>
  <PROPERTY NAME="REPOSITORYSERVICE">
    <BEAN
CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMREPOSITORYSERVICEIMPL" />
  </PROPERTY>
  <PROPERTY NAME="CLOCK" REF="ACTIVITICLOCK" />
  <PROPERTY NAME="CUSTOMSESSIONFACTORIES">
    <LIST>
      <BEAN
CLASS="COM.GKOTSCHY.ACM.ACTIVITI.ACMHISTORYMANAGERSESSIONFACTORY" />
    </LIST>
  </PROPERTY>
  <PROPERTY NAME="CUSTOMFORMTYPES">
    <LIST>
      <BEAN CLASS="COM.GKOTSCHY.ACM.ACTIVITI.TEXTFORMTYPE" />
    </LIST>
  </PROPERTY>
  <PROPERTY NAME="CONFIGURATORS">
    <LIST>

```

```

        <REF BEAN="LDAPCONFIGURATOR" />
    </LIST>
</PROPERTY>
</BEAN>
<BEAN ID="PROCESSENGINE"
CLASS="ORG.ACTIVITI.SPRING.PROCESSENGINEFACTORYBEAN">
    <PROPERTY NAME="PROCESSENGINECONFIGURATION"
REF="PROCESSENGINECONFIGURATION" />
</BEAN>
...
</BEANS>

```

Table 13 Excerpt from root context of ACM prototype

The root context excerpt above shows the configuration of the Activiti process execution engine. First the Activiti services (described in chapter 5.2.3.1) are created using the Factory software design pattern [60], as well as a clock reader that serves as a time source for the process execution engine. This is followed by the process execution engine configuration, which starts with referencing the data source and transaction manager declared in the persistence layer. Job execution is deactivated as it is not relevant for the topics discussed in this thesis, and the Activiti history component is configured to keep a full record of all changes to process entities. Next a sample process model is setup for auto-deployment, by specifying a source, name, category and tenant, as well as the responsibilities Starter (the application itself) and Owner (a group of users defined in the organisation hierarchy). The customised Activiti services (RuntimeService, TaskService, FormService, IdentityService and RepositoryService) are declared next, and a customised history manager is created through a customised session factory. Then a sample custom form type (i.e. a non-standard attribute type used in Activiti forms) is declared, as well as the LDAP configurator described in chapter 5.2.1.2. Finally the process engine itself is created and injected with the engine configuration.

5.2.5 Presentation Layer

The presentation layer consists of the controllers responsible for assigning a model (provided by the service layer) to a view, as well as various components used to display a view. Apache Tiles is a templating framework based on the Composite software design pattern [60], which allows the definition of page fragments that can be assembled into complete pages at run-time. [79] Java ServerPages (JSP) facilitate the dynamic generation of web pages based on HTML, XML and other document types, and are the Java equivalent of other server-side scripting languages, such as PHP and ASP. [80] Finally Twitter Bootstrap is a light-weight user interface (UI) framework providing standard UI components based on HTML, CSS and JavaScript. [81]

```

<?XML VERSION="1.0" ENCODING="UTF-8"?>
<BEANS:BEANS XMLNS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC"
XMLNS:BEANS="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS"
XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
XSI:SCHEMALOCATION="HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/MVC/SPRING-MVC.XSD
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS
HTTP://WWW.SPRINGFRAMEWORK.ORG/SCHEMA/BEANS/SPRING-BEANS.XSD">
...
<!-- TILES -->

```

```

<BEANS:BEAN ID="TILESVIEWRESOLVER"
CLASS="ORG.SPRINGFRAMEWORK.WEB.SERVLET.VIEW.TILES3.TILESVIEWRESOLVER">
  <BEANS:PROPERTY NAME="ORDER" VALUE="0" />
</BEANS:BEAN>
<BEANS:BEAN ID="TILESCONFIGURER"
CLASS="ORG.SPRINGFRAMEWORK.WEB.SERVLET.VIEW.TILES3.TILESCONFIGURER">
  <BEANS:PROPERTY NAME="DEFINITIONS">
    <BEANS:LIST>
      <BEANS:VALUE>/WEB-INF/TILES/ACM-TILES.XML</BEANS:VALUE>
    </BEANS:LIST>
  </BEANS:PROPERTY>
</BEANS:BEAN>
...
</BEANS:BEANS>

```

Table 14 Excerpt from servlet context of ACM prototype

The servlet context excerpt shown above completes the Spring configuration of the ACM prototype presented in this chapter. A Tiles view resolver with highest priority is added to the servlet context, and the XML file containing the defined tiles is referenced.

```

<?XML VERSION="1.0" ENCODING="UTF-8"?>
<!DOCTYPE TILES-DEFINITIONS PUBLIC "-//APACHE SOFTWARE FOUNDATION//DTD TILES
CONFIGURATION 3.0//EN" "HTTP://TILES.APACHE.ORG/DTDS/TILES-CONFIG_3_0.DTD">
<TILES-DEFINITIONS>
  ...
  <DEFINITION NAME="FLOWS/CREATE-ATTACHMENT/SELECTSOURCE" TEMPLATE="/WEB-
INF/VIEWS/LAYOUT-DEFAULT.JSP">
    <PUT-ATTRIBUTE NAME="TILESTITLE" VALUE="ACM / ATTACHMENTS / CREATE /
SELECT SOURCE" />
    <PUT-ATTRIBUTE NAME="TILESHEADER" VALUE="/WEB-INF/VIEWS/HEADER.JSP"
/>
    <PUT-ATTRIBUTE NAME="TILESBODY" VALUE="/WEB-INF/FLOWS/CREATE-
ATTACHMENT/SELECTSOURCE.JSP" />
    <PUT-ATTRIBUTE NAME="TILESFOOTER" VALUE="/WEB-INF/VIEWS/FOOTER.JSP"
/>
  </DEFINITION>
  ...
</TILES-DEFINITIONS>

```

Table 15 Excerpt from Tiles configuration of ACM prototype

The sample tile definition shown above references a base template, specifies a page title, and splits the screen into 3 basic elements. The header element contains the main navigation bar, the footer element displays basic information on the authenticated user and the execution time, and the body element holds the specific content of this page, i.e. a form allowing a user to select the source of an attachment.

6 Test Scenarios

This chapter presents various test scenarios to demonstrate the capabilities of the ACM prototype to execute unpredictable processes. Each test scenario demonstrates a modelling

pattern for choice, and identifies process elements suited to provide the flexibility offered by each of the case management approaches presented in chapter 2.7.

Business Process Management (BPM) offers events that occur during the lifecycle of a process, and can influence the workflow by suspending execution of a path until the occurrence of a certain event. Most events are based on the process state, but some rely on interaction with the outside world. The timer event waits for a pre-defined amount of time to elapse, while the message event indicates that the process is waiting for a message sent by some external actor. Signal events are similar to message events, as they suspend the active path until a certain interaction with the process is initiated from the outside. But unlike message events, signal events contain no content payload (i.e. do not affect the process state), and occur simultaneously over all active instances of a certain process model. Despite this limitation in the standard (i.e. BPMN 2.0) definition of signal events, the Activiti API offers a method to address a signal directly to a specific process instance, thus enabling a user or IT system to interact with a process at run-time.

Production Case Management (PCM) offers the flexibility to choose from a set of pre-defined solutions. In terms of BPMN modelling, this is usually expressed through the call activity element. By definition, a call activity is a static (i.e. a specific task or process is referenced at design-time) wrapper for a globally defined task or process re-used in the context of the current process. Activiti limits the implementation of call activity to referencing processes, but allows certain values of the process model (including the process referenced by a call activity) to be defined in expression language. An expression is defined at design-time, but evaluated at run-time (not just against the process state by the way, as referencing a service instead of a process variable enables the expression to access the entire application state). This allows for a scenario where the knowledge worker chooses a certain process at run-time, and the call activity acts as an execution container.

Adaptive Case Management (ACM) promises the flexibility to come up with your own plan at run-time. This requires not just relying on pre-defined solutions, but a method for modelling solutions during process execution. Activiti already tracks the association of process instances to tasks and other process instances, as both are created by process instances according to the pre-defined process model. It also tracks the association of task to task, as sub-tasks can be created by a user assigned to a user task. The missing puzzle piece is tracking the association of task to process, i.e. allowing the creation of process instances by a user assigned to a user task. The key idea is that Activiti out-of-the-box already allows users to create unstructured activities (i.e. user tasks) at execution time, and only has to be adapted to allow the non-anticipated creation of structured activities (i.e. processes).

6.1 Test Scenario 01: State-based Choice

The aim of this test scenario is to demonstrate a choice modelled in standard BPMN, which can be executed by any BPMN-compliant execution engine. The scenario consists of a user presented with the choice to either perform one of three pre-defined tasks, or perform no task at all.

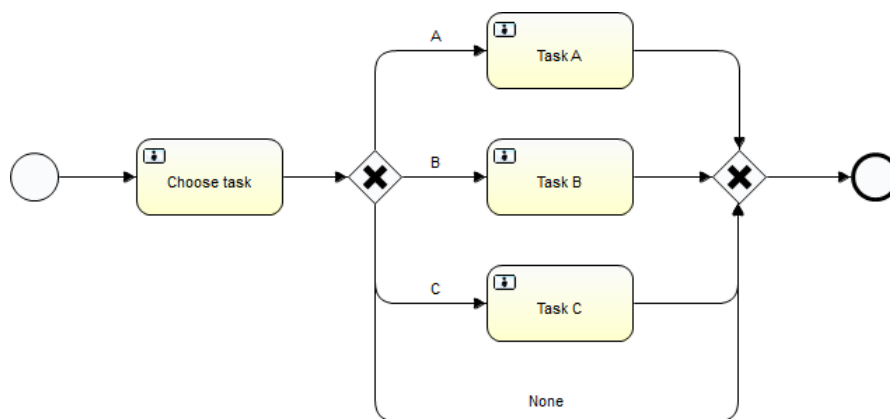


Figure 20 Activiti process model for state-based choice

As shown above, the scenario starts with a user task. To complete the task, the acting user chooses one of four options made available through a task form. Upon completion of the task, the chosen value is stored in the process state, and used to select a single outgoing transition from the succeeding exclusive gateway. The first three transitions (A, B and C) depend on a specific value having been chosen, while the fourth (None) is a default transition, i.e. it is chosen if no other transition qualifies. The process ends once the transitions merge again, i.e. if either no task was chosen or the chosen task was completed.

Id	Name	Type	Expression	Variable	Default	Pattern	Required	Readable	Writea...	Form values
choice	Choice	enum			a		true	true	true	a:Task A;b:Task B;c:Task C:none:None

Figure 21 Configuration of task form in Activiti Designer

The task form definition shown above contains a single form attribute of type Enum. Its default value is specified as “a”, while its potential values are listed as “a”, “b”, “c” and “none”. The chosen value is stored in the process state as a process variable whose name equals the Id of the form attribute. Process variables can be used in expressions, thus allowing each transition to check its condition against the current process state.

The ACM prototype can successfully complete this test scenario. The process model only relies on standard BPMN concepts, and the Activiti component does not need to be customised for this test scenario as it supports all applied elements natively.

6.2 Test Scenario 02: Signal-based Choice

The aim of this test scenario is to demonstrate a choice based on process signals. Standard BPMN does not support the concept of signalling specific process instances, but the Activiti API offers an appropriate functionality. Similar to the previous scenario, this scenario consists of a user presented with the choice to either perform one of three pre-defined tasks. However the user is also offered the choice of cancelling the choice at any time, i.e. also after a task is chosen but before it is completed.

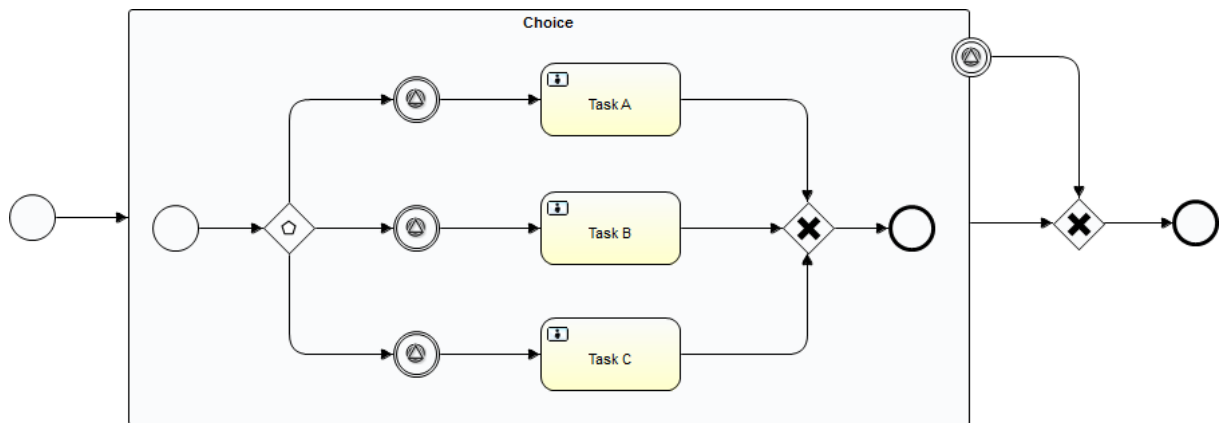


Figure 22 Activiti process model for signal-based choice

As shown above, the scenario starts with a subprocess, which contains a boundary signal allowing the user to cancel the subprocess and end the main process immediately. The subprocess begins with an event-based gateway, which makes three signals available. If any of these signals is activated by the user, its succeeding task (Task A, Task B or Task C) is created and all three signals become unavailable. Once the task is completed, both the subprocess and finally the process end.

Id	Name	Scope
task_a	task_a	global
task_b	task_b	global
task_c	task_c	global
cancel	cancel	global

Figure 23 Configuration of process signals in Activiti Designer

The process signals definition shown above contains three signals (task_a, task_b and task_c) that create a corresponding user task, as well as a fourth signal (cancel) that terminates the subprocess. While the Activiti API supports the activation of process signals in specific process instances, it lacks a concept for controlling access to those signals. Based on the collaborator lifecycle (discussed in chapter 7.5), the ACM prototype restricts access to process signals to the process instance owner (e.g. the case manager) and assignee (e.g. the case handler).

The ACM prototype can successfully complete this test scenario. The process model relies on non-standard BPMN concepts (instance-specific process signals), but the Activiti component supports this concept, and the ACM prototype adds access control to ensure that signals are only activated by authorised users.

6.3 Test Scenario 03: Repetitive Choice

The aim of this test scenario is to demonstrate a repetitive choice based on process signals. Similar to the previous scenario, the user is presented with a choice of performing three tasks. However in this scenario once a user makes the choice to create a task, he is immediately able to choose again (i.e. before completing the chosen task), thus allowing for the dynamic creation of any number of tasks that have been specified at design-time.

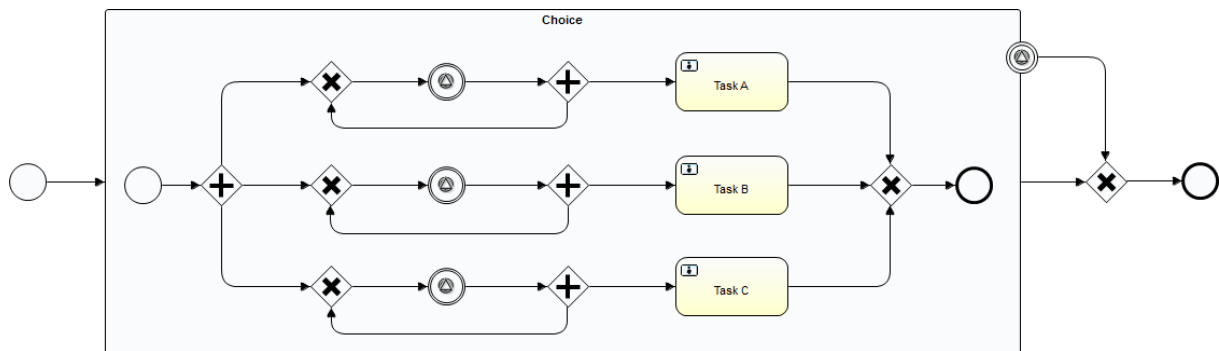


Figure 24 Activiti process model for repetitive choice

As shown above, the scenario starts with a subprocess, which contains a boundary signal allowing the user to cancel the subprocess and end the main process immediately. The subprocess begins with a parallel gateway, which makes three signals available. If any of these signals is activated by the user, another parallel gateway splits the execution and creates the selected task, as well as re-activating the signal. The choice to create a pre-defined task remains active, until the user chooses to cancel the subprocess, and thus end the process (and all non-completed tasks created by it).

Process Instances / TS03

TS03
Active

ID 36

Started 15/10/2017 18:12

Starter Georg Kotschy

Tenant TU Wien

Process Definition Thesis Test Scenario 03

Suspended false

Ended false

Suspend
 Delete
 Comment
 Attachment
 Choose A
 Choose B
 Choose C
 Cancel choice

Comments

No comments found.

Sub Processes

No sub processes found.

Tasks 3 1

Name	Category	Deadline	State	Tenant
Task A	None	None	Ended	TU Wien
Task B	None	None	Unassigned	TU Wien
Task C	None	None	Unassigned	TU Wien
Task A	None	None	Unassigned	TU Wien

Figure 25 Screenshot of process instance UI showing repetitive choice

The screenshot above shows this test scenario being executed in the ACM prototype. The user has four choices (Choose A, Choose B, Choose C and Cancel choice) available, and the task list shows that several tasks have previously been created (and one of them

completed). While this pattern effectively demonstrates freedom of choice for the user to complete any, all or none of the selected tasks, it depends on pre-defined tasks and does not allow the user to create and select and new task at run-time.

The ACM prototype can successfully complete this test scenario. The process model relies on non-standard BPMN concepts (instance-specific process signals), but the Activiti component supports this concept, and the ACM prototype adds access control to ensure that signals are only activated by authorised users.

6.4 Test Scenario 04: Run-time Choice

The aim of this test scenario is to demonstrate choices made at run-time, i.e. choices involving process elements that were not defined before the process instance was created.

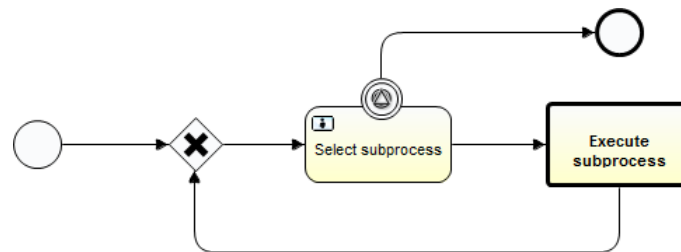


Figure 26 Activiti process model for run-time choice

As shown above, the scenario starts with a user task, allowing the acting user to specify a process definition. The selection is resolved as an expression when the following call activity is created, and the specified process definition is executed as called element. Similarly to previous scenarios, the user can cancel the selection, by activating a signal. In this case the boundary signal is contained by the user task itself, instead of a surrounding subprocess. This cancellation pattern only allows for the cancellation of the subprocess selection, not the subprocess execution.

Process Instances / TS04 / Execute subprocess

TS04 / Execute subprocess Active

ID 43

Started 15/10/2017 21:02

Starter Georg Kolschy

Tenant TU Wien

Process Definition Thesis Test Scenario 01

Super Process TS04

Suspended false

Ended false

Comment

Attachment

Comments

No comments found.

Sub Processes

No sub processes found.

Tasks 1

Name	Category	Deadline	State	Tenant
Choose task	None	None	Assigned	TU Wien

Figure 27 Screenshot of process instance UI showing subprocess chosen at run-time

The screenshot above shows this test scenario being executed in the ACM prototype. The subprocess (TS04 / Execute subprocess), based on the process definition selected by the user (Thesis Test Scenario 01), has been launched by a call activity element in the main process (TS04). The specific process definition used to launch the call activity is resolved at run-time, i.e. the user can choose any process definition existing at the moment in time when the call activity element is created, and thus after the main process instance is created. As discussed in the description of the chosen process definition in chapter 6.1, the launched subprocess has created a user task (Choose task) for demonstrating state-based choice.

The ACM prototype can successfully complete this test scenario. The process model relies on non-standard BPMN concepts (instance-specific process signals and call elements specified by expression), but the Activiti component supports both concepts, and the ACM prototype adds access control to ensure that signals are only activated by authorised users.

6.5 Test Scenario 05: Process Migration

The aim of this test scenario is to demonstrate the migration of running process instances. The modelling patterns demonstrated in the previous test scenarios can be re-composed to allow a user to make task- or signal-based repetitive choices at run-time. The way the choice is made (i.e. the process definition of the main process itself) however is modelled at design-time, and standard BPM does not address the possibility of changing the process definition of a running process instance. This scenario consists of two process definitions, which can be used to migrate a running process instance (based on the first process definition) to a newer version (based on the second process definition).

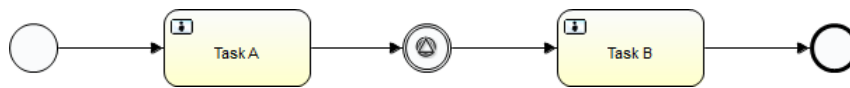


Figure 28 Activiti process model for migration (version 1)

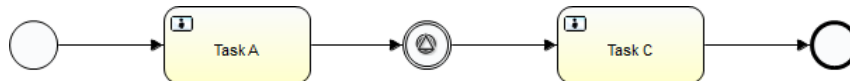



Figure 29 Activiti process model for migration (version 2)

As shown above, two versions of the same process have been modelled. Each process starts out by creating Task A, and (upon completion of Task A) making a signal available to the user. In version 1 of the process definition Task B is created once the user activates the available signal. However if the running process instance is migrated to version 2 before the signal is activated, Task C will be created instead.

Process Instances / TS05 / Migrate

TS05 Suspended


 ID 28
 Started 15/10/2017 20:26
 Starter Georg Kotschy
 Tenant TU Wien
 Process Definition Thesis Test Scenario 05
 Suspended true
 Ended false

Please select version

Process Definition Version 1 (current)

Version 2
Version 1 (current)

Figure 30 Screenshot of process instance UI showing migration

As shown in the screenshot above, a running process instance has been suspended. The migration user interface allows a process owner to change the underlying process definition, by selecting a different version. The caveat for process migration using Activiti is that the past must be identical in both versions. More specifically, every element (of the older process definition) that has been active in the past (and thus recorded by the Activiti history component), has to be present in the newer version of the process definition. Once the process instance has been migrated and activated, execution continues with the same process elements active before suspension, but with the potential for a different future.

The ACM prototype can successfully complete this test scenario. The process model relies on non-standard BPMN concepts (instance-specific process signals and process instance migration), but the Activiti component supports both concepts, and the ACM prototype adds access control to ensure that signal activation and instance migration are only performed by authorised users.

6.6 Test Scenario 06: Task Templates

The aim of this test scenario is to demonstrate the adaption of a user task as case container, able to contain further user tasks and process instances. While the creation of tasks and

subprocesses by process instances, as well as of subtasks by tasks, is supported by both BPMN and Activiti, several other capabilities are required for this test scenario: an association between tasks and process instances allowing for the creation of process instances in the context of a user task; a task template management system for patterns of case activities (i.e. tasks and process instances) defined at run-time; and a set of logical and temporal rules for specifying goals, completion conditions, and relationships between case activities. Due to the extensive customisation required, and the time limitation imposed for this thesis, the ACM prototype is not able to support these capabilities. However this test scenario is defined to indicate a promising area for further research.

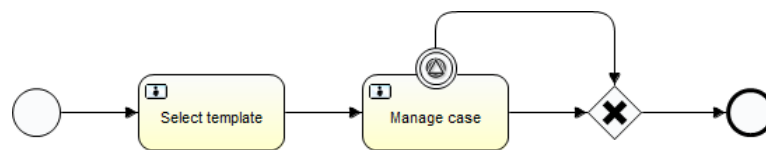


Figure 31 Activiti process model for task templates

In the process model shown above, a user task is created, which allows the acting user to select a task template. This is followed by another user task, acting as a case container and based on the selected task template. All case activities are created in the context of this user task, and case templates can be used to create a pre-defined pattern of case activities contained by the task. Each of these case activities is again a task or process, allowing for the recursive execution of structured and unstructured processes. As case templates are managed outside of the execution engine, they do not need to be defined in BPMN, and can rely on application-specific logical and temporal rules. The completion of such a user task depends on goals, which are formulated using the logical and temporal rules implemented by the application. Goal achievement may depend either on human judgement (e.g. by the case handler himself, or his case manager confirming that the content of a provided document is satisfactory) or automated business logic (e.g. checking whether a document has been provided, regardless of its content). As in previous test scenarios, a signal event is created to allow the user to end a case prematurely.

```

...
<!-- TASK INSERT -->
<INSERT ID="INSERTTASK"
PARAMETERTYPE="ORG.ACTIVITI.ENGINE.IMPL.PERSISTENCE.ENTITY.TASKENTITY">
  INSERT INTO ${PREFIX}ACT_RU_TASK (ID_, REV_, NAME_, PARENT_TASK_ID_,
DESCRIPTION_, PRIORITY_, CREATE_TIME_, OWNER_,
      ASSIGNEE_, DELEGATION_, EXECUTION_ID_, PROC_INST_ID_,
PROC_DEF_ID_, TASK_DEF_KEY_, DUE_DATE_, CATEGORY_, SUSPENSION_STATE_, TENANT_ID_)
  VALUES ({ID, JDBCTYPE=VARCHAR},
      1,
      #{NAME, JDBCTYPE=VARCHAR},
      #{PARENTTASKID, JDBCTYPE=VARCHAR},
      #{DESCRIPTION, JDBCTYPE=VARCHAR},
      #{PRIORITY, JDBCTYPE=INTEGER},
      #{CREATETIME, JDBCTYPE=TIMESTAMP},
      #{OWNER, JDBCTYPE=VARCHAR},
      #{ASSIGNEE, JDBCTYPE=VARCHAR},
      #{DELEGATIONSTATESTRING, JDBCTYPE=VARCHAR},
      #{EXECUTIONID, JDBCTYPE=VARCHAR},
      #{PROCESSINSTANCEID, JDBCTYPE=VARCHAR},

```

```

        #{PROCESSDEFINITIONID, JDBCTYPE=VARCHAR},
        #{TASKDEFINITIONKEY, JDBCTYPE=VARCHAR},
        #{DUEDATE, JDBCTYPE=TIMESTAMP},
        #{CATEGORY, JDBCTYPE=VARCHAR},
        #{SUSPENSIONSTATE, JDBCTYPE=INTEGER},
        #{TENANTID, JDBCTYPE=VARCHAR}
    )
</INSERT>
...

```

Table 16 Excerpt from Activiti task query definition

As shown in the Activiti task query definition excerpt above, a newly created task can be associated to another task (via `PARENT_TASK_ID_`) or a process instance (via `PROC_INST_ID_`). Processes instances however lack the possibility to specify an associated task, thus preventing a process instance from being created in the context of an existing user task. This missing relation is essential to enabling a recursive run-time definition of tasks and processes in the context of a user task acting as case container. The case activities defined during execution could be re-used and shared between users via a task template management system provided by the IT application. As unstructured processes lack an explicit sequence of activities, the IT application could also provide a custom set of logical and temporal rules. These are required to impose some level of order on the case activities, as well as to define goals for task, and measure the completion of those goals.

The ACM prototype cannot complete this test scenario. The extensive customisation required to support this scenario is not possible due to the limited time available for completing this thesis, but the level of flexibility provided by this approach makes it an excellent candidate for further research.

7 Evaluation

This chapter evaluates the capabilities of the ACM prototype in regard to the core requirements selected in chapter 3.4. While the main focus of the prototype lies on investigating the execution of unpredictable processes, significant work has also been done on controlling access to process entities.

7.1 Interoperability


Open source software greatly relies on interoperability to deliver comprehensive solutions. While vendor-based software attempts to cover the entire spectrum of case management capabilities, often building process execution engines on top of existing content management components, open source-based solutions have to integrate various heterogeneous components to deliver a similar range of functionality.

The ACM prototype is designed to maximise interoperability. Object-oriented Programming (OOP) languages support the re-use of information and business logic within an application component. The chosen programming language Java furthermore ensures that the server-side application is platform-independent and can be deployed on various operating systems. The prototype is implemented as a web application, thus ensuring that the client-side is platform-independent as well. The user interface requires a modern web browser, but the responsive UI works on mobile devices as well as personal computers. Though a specific web server was chosen for development, the Java Servlet standard allows deployment on a

variety of compliant web servers. The chosen application framework provides an Inversion of Control (IOC) container, and facilitates the exposure and re-use of business logic over a network as web services [82], based on two popular standards: Simple Object Access Protocol (SOAP) for exchanging structured information via XML, based on a messaging envelope structure designed to carry control information in the message header and application payload in the message body [83]; and Representational State Transfer (REST), which allows requesting IT systems to access and modify resources over a network using a uniform and pre-defined set of stateless operations [75]. The service-oriented application architecture defines 5 distinct layers of responsibility, and relies on two standards to access data sources and directories: Java Database Connectivity (JDBC), an API allowing for the programmatic execution of SQL statements, retrieval of results, and propagation of changes back to an underlying data source [71]; and Lightweight Directory Access Protocol (LDAP), an open industry standard for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network [62]. The chosen process execution engine applies the Business Process Model and Notation (BPMN) standard for process modelling, though execution engine-specific process elements limit model interoperability with other process execution engines. And while a simpler pattern for handling content was implemented instead, the Content Management Interoperability Services (CMIS) define a standardised abstraction layer allowing full control over the lifecycle of content stored in an external document or record management system. [84]

Groups / Adaptive Case Management

Adaptive Case Management



Type Organisation
Distinguished Name ou=acm,ou=groups,dc=gkotschy,dc=com

Sub Groups

Name	ID
Roles	roles
Organisations	organisations

Roles

Name	ID
Administrator	roles:administrator
Student	organisations.at.ac.tuwien:student
Professor	organisations.at.ac.tuwien.informatics.isis.ecommerce:professor

Members

Name	Email	Roles	Tenant
ACM Application	acm@gmail.com	Administrator	TU Wien
Georg Kotschy	gkotschy@gmail.com	Student	TU Wien
Jürgen Dorn	juergen.dorn@ec.tuwien.ac.at	Professor	TU Wien

Hello Georg Kotschy, you are signed in as kotschge for TU Wien without any authorities.
20/10/2017 18:33 (78ms)

Figure 32 Screenshot of group management UI

The screenshot above shows the group management user interface of the ACM prototype. The root node is displayed on top, followed by a list of successor nodes (i.e. sub groups), a list of roles contained by the currently viewed node, and a list of users associated to the currently viewed node by one of these roles. Roles are a flat table of elements with a unique common name, while organisations are a recursive hierarchical structure based on a unique namespace. All three concepts are stored on a directory server, accessed via the LDAP protocol, and displayed in the UI of the ACM prototype.

As required in chapter 3.4.1, the ACM prototype makes extensive use of the existing standards described above, to facilitate interaction with external IT systems.

7.2 Process Execution

Most processes contain process fragments that can be fully automated, or are at least fully predictable, even if they require human interaction. Rather than providing knowledge workers with a system that can only execute unstructured processes, the need to execute processes with an arbitrary level of structure should be addressed. While there are currently no standard solutions available for modelling and executing adaptive processes, standard solutions do exist for both business process modelling and execution. Therefore, the ACM prototype relies heavily on an execution engine for structured processes, which is based on a standard modelling notation. As demonstrated extensively in chapter 6, Activiti fulfils both criteria, but lacks any built-in support for executing adaptive processes. As discussed in chapter 6.6, various customisations of the core Activiti engine would be required to handle the unpredictability of adaptive processes.

The uni-directional association specified by BPMN only allows the pre-defined creation of tasks by process instances, and the ad-hoc creation of tasks associated with other tasks. A bi-directional association of tasks and processes enables case workers to add both unstructured tasks and structured processes to a process instance at run-time. Task templates provide re-usable patterns for unstructured tasks as containers of other tasks or processes. Similar to process template management, task template management depends on the user community to identify, review, recommend and re-use patterns that have proven successful in past cases. Using a single unstructured task as the root element of a process allows for full flexibility at run-time. While the initial content would be based on a task template, such a root element would only be bound to a directly preceding start event, and a directly succeeding end event. Given the recursive nature of tasks, and a bi-directional association of tasks and processes, this root element can act as a container for all structured (e.g. process) and unstructured (e.g. task) elements of a case.

As required in chapter 3.4.2, the ACM prototype demonstrates the level of unpredictability supported by the chosen process execution engine. Furthermore the research conducted for this thesis indicates that extensive customisation of Activiti could allow for the use of a user task as an unstructured container for case activities, based on task templates managed outside of the execution engine.

7.3 Decision Making

Processes rely on decision making to determine the appropriate sequence of activities. More specifically, outgoing transitions may include an activation condition, allowing the process to determine at run-time which outgoing transitions should be followed, and which should be ignored. Regardless of the decision source, the basic pattern for achieving this effect is

always the same. The transition condition is formulated as an expression that references a value available in the context, which includes process variables stored in the process state, and application services deployed in the IoC container. The expression evaluates to true or false, and usually either compares (e.g. equals, smaller than, or greater than) the value against a constant defined in the process model at design-time, or relies on business logic implemented outside of the process to directly determine a boolean value. Process variables can be injected into the process state by various process elements: a user task containing a form to be filled out by a human user; a Java service task invoking a service deployed in the IoC container; a web service task synchronously accessing an external web service; or a business rule task synchronously executing one or more rules in an external rules engine.

The full range of decision making capabilities of the chosen process execution engine is available in the ACM prototype. Based on the Service-Oriented Architecture (SOA) design pattern Rules Centralisation, the business logic required for decision making is split over two service models: Business Activity Services that encapsulate business logic applicable to more than one process model; and Process Services containing business logic for a specific process model. While this design pattern facilitates re-use by specifying rules logic as services (and thus outside of the process model), it impedes change being effected through the process model. Therefore it should not be applied universally for all decision making in a process, but rather reserved for repeatedly occurring non-trivial decisions.

Structured processes modelled with BPMN use transitions to define a specific physical ordering of process elements at design-time. An unstructured process (e.g. the enhancement of user task as discussed in chapter 6.6) requires a different approach, as physical sequences are not sufficient for ordering an unpredictable sequence of activities. The Case Management Model and Notation (CMMN) presented in chapter 2.8.2, offers various elements for such purposes, such as milestones, sentries and timer events. While no standard for CMMN execution exists, and existing process execution solutions claiming support for CMMN only cover a fraction of CMMN elements, the behaviour defined by CMMN can be implemented in custom application logic and applied to the case activities contained by an unstructured process element like a user task. Completion of a user task would thus depend on progress evaluation of the goals defined for the task, and custom rules specifying the temporal or semi-sequential order of case activities. Unlike a design-time model based on BPMN or CMMN, custom application logic can allow for run-time changes, and capture such changes for future re-use.

As required in chapter 3.4.3, the ACM prototype provides the capability to make decisions both based on the process state and based on the application state. Furthermore the research conducted for this thesis indicates that non-explicit ordering of case activities contained in an unstructured process, as well as a definition of process completion goals can be achieved through custom application logic similar to CMMN.

7.4 Content Management

As discussed in chapter 3.1.4, case management is data- and document-centric, and associating content with process elements is essential to deriving meaning from either. Well-defined data structures can be stored in a relational database, while semi-structured and unstructured information can be stored in a document-oriented database. Either option allows data to be stored in an external component. Another content storage option is an external document or record management system that provides versioning and access management capabilities. Many vendor-based process execution solutions are in fact based

on existing content management solutions, and standards like Content Management Interoperability Services (CMIS) enable the complete integration of the document management lifecycle into the case management system. Finally large organisations like the EU (e.g. EUR-Lex) or European Commission (e.g. ARES) maintain organisation-wide record management systems, containing all legally relevant official documents produced by the organisation. Their capabilities are very similar to document management systems (and can therefore be integrated in a similar manner), however the record management lifecycle prevents the alteration of records after storage.

The persistence layer of the ACM prototype stores data in a database, and provides services access to that data through Data Access Objects (DAO). A reference to data managed by the persistence layer has to be stored in the process state, and can be passed along in service calls or used to display the data in the user interface.

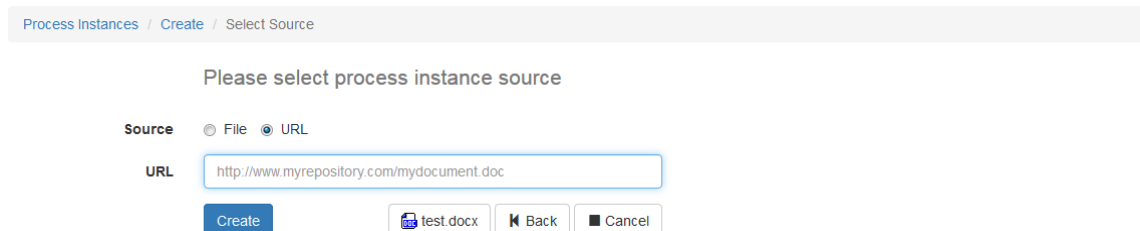


Figure 33 Screenshot of create attachment UI

The prototype relies on the built-in capabilities of Activiti for attaching documents to process entities. As shown in the screenshot above, attachments can be uploaded through the web application, or referenced by a URL pointing to the storage location. In either case the process layer stores an explicit association between the attachment (i.e. uploaded file or referenced URL) and the process entity (i.e. user task or process instance), and access control is enforced through the collaborator lifecycle (see chapter 7.5).

As required in chapter 3.4.4, the ACM prototype demonstrates its ability to associate content with process elements. Furthermore the full integration of an external document and record management system via a standard protocol such as CMIS is possible, but requires extensive coding to implement all available operations, and to provide a user interface allowing the user to perform those operations.

7.5 Identity and Access Management

Authorisation associates identified users (or groups of users) with specific resources like application components (e.g. upload a document) and business objects (e.g. documents, processes, tasks). This limits the user's possible interactions with the case management system, and is essential for providing information security. The ACM prototype supports various concepts for access control, some of which require extensive customisation of the underlying process execution engine.

A directory server acts as authentication provider, containing the identity records of all users including usernames and passwords required to identify an acting user. By using a standard protocol for accessing and maintaining this information, the case management system stays independent of the choice of authentication provider. Activiti does not just offer native support for resolving authorities against a directory server, but also provides an API to access the information stored on the directory server.

Assignee	<input type="text" value="{authenticatedUserId}"/>
Candidate user...mma separated)	<input type="text"/>
Candidate grou...mma separated)	<input type="text" value="organisations.at.ac.tuwien"/>
Form key	<input type="text"/>
Due date (variable)	<input type="text"/>
Priority	<input type="text"/>
Category	<input type="text"/>
Skip expression	<input type="text"/>

Figure 34 Configuration of user task in Activiti Designer

As shown in the screenshot above, interactive process elements like user tasks can be associated with a user responsible for completion of the task (i.e. assignee). This association can consist either of a specific username determined at run-time (usually after a user claims an unassigned task), or of an expression specified at design-time. In the example above, once the user task is created during process execution, it will be assigned to the acting user, i.e. the last user who interacted with the process and caused this process element to be created. While the assignee attribute specifies a single person responsible for a process element, candidate users or groups define a set of users eligible for taking responsibility over a process element. In the example above, any user associated with the specified group (i.e. organisations.at.ac.tuwien) is eligible.

Both options discussed above are a form of resource-based access control, as a specific resource (e.g. user task) is associated with an authority (i.e. user or group) in a specific context (e.g. assignee or candidate). However the directory server also contains a list of roles, which can be used for enforcing functional access control in the application, e.g. by requiring any user who claims or completes a task to have a specific role (e.g. case handler) assigned to them.

Tenancy is an access control concept that was newly introduced in the version of Activiti used by the ACM prototype. Data is partitioned according to tenant organizations, and no tenant can see the models or definitions of other tenants. Unlike the concepts discussed above, tenancy requires not just to prevent unauthorised access to a resource or functionality, but rather to prevent knowledge of that resource's existence in the first place. This means tenancy has to be implemented at the lowest levels, assuring that any query for process elements only returns elements visible by the tenant organisation of the acting user.

While Activiti natively supports specifying assignees and candidates for user tasks, as well as candidates for process definitions (i.e. indicating which users may create a process instance from a certain process definition), other semantic relationships with process elements need to be managed as well. The Activiti component of the ACM prototype has been customised extensively, to allow for the management of semantic relationships representing a collaboration between users or groups with a process model, process definition, process instance or task.

	Process Model	Process Definition	Process Instance	Task		
Candidate		+ create model	✓	+ create task	✗	
		± definition.collaborator	✗	± task.collaborator	✗	
		users/groups	M	users/groups	M	
Starter	+ create model	✓	+ create instance	✓	+ create task	□

	user	I		user	I	user	I
Owner	+ create model	✓		+ create instance	✓	+ create task	□
	± model.collaborator	✗		± instance.collaborator	✗	+ claim task	✓
						- unclaim task	✓
	user/group	T		user/group	T	user	I
Assignee				+ create instance	✗	+ create task	□
				± instance.collaborator	✗	+ claim task	✓
						- unclaim task	✓
						± reclaim task	✓
						± delegate task	✓
						± resolve task	✓
				user/group	T	user	I
Participant				+ claim task	✓		
				+ delegate task	✓	+ resolve task	□
				+ resolve task	✓		
				+ complete task	✓		
				users	I	users	I
Custom				± instance.collaborator	✗	± task.collaborator	✗
				users/groups	M	users/groups	M

✓ ...implemented, □ ...to-do, ✗ ...out-of-scope, I...immutable, T...transferable, M...mutable, {optional}, {entity}:{collaboration}, {entity}:{attribute}

Table 17 Collaborator lifecycle for process entities

As shown in the table above, four types of process entities are associated with six types of semantic relationships. The associated authorities can be users and/or groups, and the relationship itself can be immutable (i.e. it is created automatically, as the result of a certain action, and cannot be transferred to another authority), mutable (i.e. it is created manually), and transferable (i.e. it is created automatically, but can be transferred manually to a different authority). The state of implementation in the ACM prototype is indicated for every action, and due to time limitations, some actions were not implemented.

The ACM prototype extensively demonstrates access control based on tenants, users, groups and roles. Furthermore, the collaborator lifecycle proposed in this thesis serves to define authorised interactions with all process entities.

7.6 Accountability and Transparency

As discussed in chapter 3.4.6, accountability and transparency serve the organisational goals of openness and generation of trust. Accountability requires the keeping of detailed historical records regarding who did what, when, and to which resource. The ACM prototype has a single persistence layer, which can be manipulated through either Data Access Objects (DAO) or the Activiti API. DAOs depend on Hibernate as database abstraction layer, which offers its own easy-to-use auditing component. Hibernate Envers allows for tracking all changes to data entities effected through Hibernate, i.e. all data modifications performed through DAOs. It does not however include process entities, which are modified directly by Activiti using native database queries. For this purpose Activiti provides its own historical data component, in the form of Activiti History, which captures and permanently stores everything that happens during process execution.

Transparency addresses the need to keep all stakeholders informed of relevant information regarding the current state of the case. This includes the presentation of collected historical information in the UI, as well as the generation of a personalised dashboard.

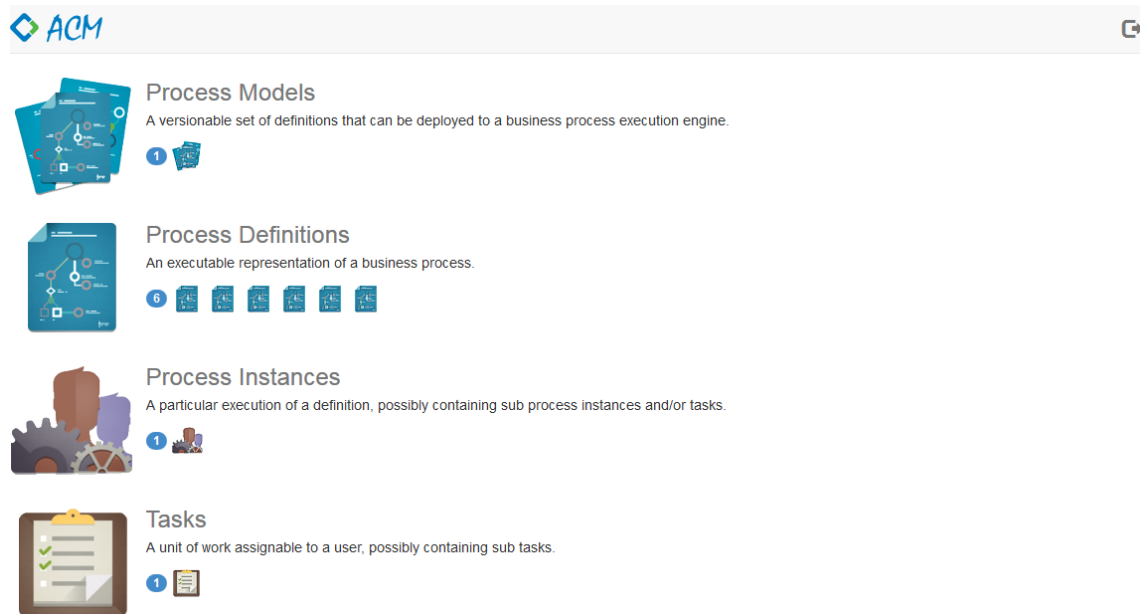


Figure 35 Screenshot of personalised dashboard

The personalised dashboard of the ACM prototype provides the user with quick access to the process entities that are associated with him via the collaborator lifecycle (see chapter 7.5). As shown in the screenshot above, a single process model is deployed containing 6 process definitions. A process instance has been created from one of these process definitions, and a task is currently pending and assigned to the user.

As required in chapter 3.4.6, the ACM prototype keeps a historical record of the process state, as well as of all business objects managed outside the execution engine. Furthermore the information collected during process execution is presented, and both the UI components and data retrieval capabilities required to build a personalised dashboard are provided.

8 Conclusion

As discussed in chapter 4.2, some business domains are inherently unpredictable. Based on the post-mortem analysis of the NEST prototype developed at the Directorate-General for Trade in 2013, the aim of this thesis is to identify a case management approach suitable for the domain of trade negotiations, as well as an open source-based technology stack able to support the chosen approach.

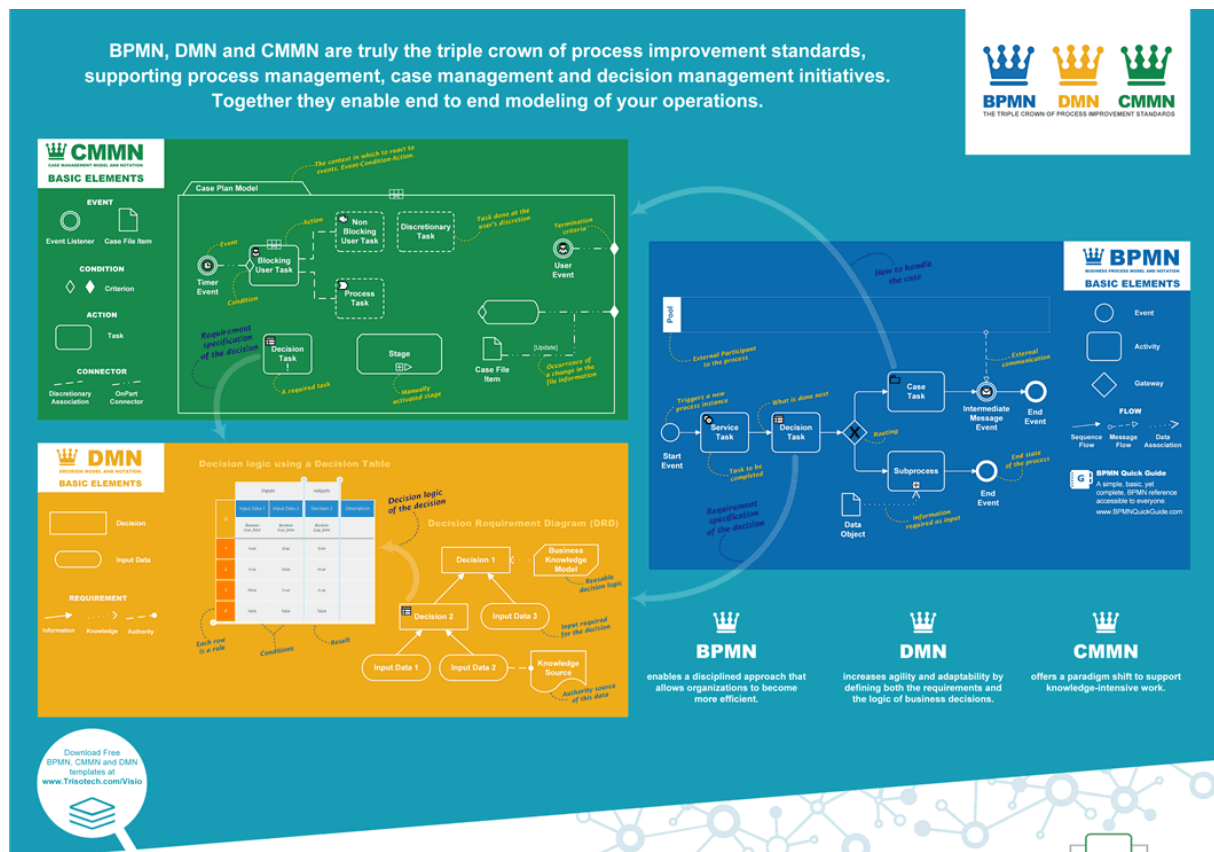
Chapter 2.7 examines various approaches to case management. Based on a comparison of three selected approaches – Business Process Management (BPM), Production Case Management (PCM), and Adaptive Case Management (ACM) – it can be concluded that ACM is the most suitable approach for unpredictable business domains such as trade negotiations. ACM supports emergent processes, i.e. processes that unfold step-by-step, with each step yielding knowledge that may influence the execution of subsequent steps. This run-time flexibility empowers knowledge workers to plan their case as it happens, and

adapt to changing circumstances that could not be foreseen at design-time. However the focus of ACM on run-time design is not supported by existing process execution engines that are largely based on BPM. An ACM system therefore requires extensive customisation of an existing process execution engine, to support the execution of both predictable and unpredictable process models.

The ACM prototype developed for this thesis is based entirely on open source software. As described in detail in chapter 5, the prototype consists of numerous independent components and technologies, each providing a piece of the puzzle that represents a complete ACM system. In chapter 3.4 various core requirements of ACM are selected to limit the scope of the prototype, and an emphasis on interoperability allows for the re-use of existing systems to meet non-selected requirements. The implementation effort of an open source-based solution is quite considerable, as many of the built-in capabilities of vendor-based solutions have to be developed from scratch. The maturity of an organisation's application landscape is thus a decisive factor for choosing an open source-based approach, as newly developed applications greatly benefit from the re-use of existing IT systems providing required capabilities.

As shown by the tests conducted in chapter 6, Activiti supports a certain amount of unpredictability natively, but needs to be heavily customised to allow for the execution of truly adaptive processes. As suggested in chapter 7.2, an existing process element (i.e. user task) could serve as an unstructured container for case activities. This requires elaborate custom application logic to manage task templates, and provide support for the kind of unstructured relationships between case activities described by the Case Management Model and Notation (CMMN). While Activiti offers a very promising foundation for building an IT application capable of executing adaptive processes, the required application logic is complex, and maintaining a customised fork of an open source project like Activiti is in itself no mean feat.

Since the research for this thesis was concluded in late 2014, two projects (Camunda [85] and Flowable [86]) have indeed followed the approach of forking Activiti to provide a customised process execution engine. More specifically, the so-called Triple Crown of Process Improvement Standards defined by the Object Management Group has become a popular approach to case management, and both Camunda and Flowable aim to support it. [87]



Trisotech is a global leader in digital enterprise solutions, offering innovative and easy-to-use software tools that allow customers to discover, model, analyze and find insights into their digital enterprise.

Trisotech
Where strategies come to life!
www.Trisotech.com

Figure 36 Triple Crown of Process Improvement Standards [88]

While the Business Process Model and Notation (BPMN) and the Case Management Model and Notation (CMMN) have already been extensively discussed in chapter 2.8, the Decision Model and Notation (DMN) is added to increase agility and adaptability by defining both the requirements and the logic of business decisions. [88] Together these 3 standards currently seem to indicate that a consensus for case management has finally been reached, at least in regard to case modelling. It remains to be seen whether open source process execution engines will be able to fully support all 3 standards, and provide knowledge workers with the run-time flexibility required to manage adaptive processes.

Bibliography

- [1] Science Buddies, "Steps of the Scientific Method," [Online]. Available: <https://www.sciencebuddies.org/science-fair-projects/science-fair/steps-of-the-scientific-method>. [Accessed 12 10 2010].
- [2] G. Anthes, "Open Source Software No Longer Optional," *Communications of the ACM*, vol. 50, no. 8, pp. 15-17, 2016.
- [3] Ł. Osuszek and S. Stanek, "Case Based Reasoning as an Element of Case Processing in Adaptive Case Management Systems," *ACSIS*, vol. 6, no. Position Papers of the Federated Conference on Computer Science and Information Systems, pp. 217-223, 2015.
- [4] K. D. Swenson, "The Nature of Knowledge Work," in *Mastering the Unpredictable*, Tampa (FL), USA, Meghan-Kiffer Press, 2010, pp. 5-27.
- [5] K. D. Swenson, *How Knowledge Workers Get Things Done*, Lighthouse Point (FL), USA: Future Strategies Inc., 2012.
- [6] M. Kurz, W. Schmidt, A. Fleischmann and M. Lederer, "Leveraging CMMN for ACM: examining the applicability of a new OMG standard for adaptive case management," in *S-BPM ONE '15 Proceedings of the 7th International Conference on Subject-Oriented Business Process Management*, Kiel, Germany, 2015.
- [7] K. D. Swenson, *Mastering the Unpredictable*, Tampa (FL), USA: Meghan-Kiffer Press, 2010.
- [8] Open Group, "SOA Reference Architecture – Business Process Layer," [Online]. Available: http://www.opengroup.org/soa/source-book/soa_refarch/p11.htm. [Accessed 13 10 2017].
- [9] C. Di Ciccio, A. Marrella and A. Russo, "Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches," *Journal on Data Semantics*, vol. 4, no. 1, p. 29–57, 2015.
- [10] N. Gronau, C. Müller and M. Uslar, "The KMDL Knowledge Management Approach: Integrating Knowledge Conversions and Business Process Modeling," *Lecture Notes in Computer Science*, vol. 3336, no. PAKM 2004: Practical Aspects of Knowledge Management, pp. 1-10, 2004.
- [11] T. Davenport, *Thinking for a Living*, Boston (MA), USA: Harvard Business School Press, 2005.
- [12] J. Ukelson, "Some Thoughts on 'Thinking for a Living'," 20 12 2009. [Online]. Available: <https://ukelson.wordpress.com/2009/12/20/some-thoughts-on-thinking-for-a-living/>.
- [13] J. Ukelson, "Unstructured, Semi-Structured and Structured Processes," 15 02 2009. [Online]. Available: <http://exeedtechnology.com/unstructured-semi-structured-and-structured-processes>.

- [14] J. Ukelson, "What is Human Process Management (HPM)?," 18 12 2008. [Online]. Available: <https://www.itworld.com/article/2782460/business-process-management/what-is-human-process-management--hpm--.html>.
- [15] J. Ukelson, "What to Do When Modeling Doesn't Work," in *Mastering the Unpredictable*, Tampa (FL), USA, Meghan-Kiffer Press, 2010, pp. 29-39.
- [16] M. Pucher, "Agile-, AdHoc-, Dynamic-, Social-, or Adaptive BPM," 30 03 2010. [Online]. Available: <https://isismjpucher.wordpress.com/2010/03/30/dynamic-vs-adaptive-bpm/>.
- [17] K. D. Swenson, "State of the Art in Case Management," Fujitsu, Sunnyvale (CA), USA, 2013.
- [18] K. D. Swenson, *Taming the Unpredictable*, Lighthouse Point (FL), USA: Future Strategies Inc., 2011.
- [19] M. J. Pucher, "The Elements of Adaptive Case Management," in *Mastering the Unpredictable*, Tampa (FL), USA, Meghan-Kiffer Press, 2010, pp. 89-134.
- [20] F. M. Kraft, "Improving Knowledge Work," in *Mastering the Unpredictable*, Tampa (FL), USA, Meghan-Kiffer Press, 2010, pp. 181-210.
- [21] M. Pucher, "The Strategic Business Benefits of Adaptive Case Management," in *How Knowledge Workers Get Things Done*, Lighthouse Point (FL), USA, Future Strategies Inc., 2012, pp. 19-37.
- [22] K. Hinkelmann and A. Pierfranceschi, "Combining Process Modelling and Case Modelling," 2014.
- [23] Signavio, "Order your free BPMN 2.0 poster," [Online]. Available: <https://www.signavio.com/contact/order-your-free-bpmn-2-0-poster/>. [Accessed 01 10 2017].
- [24] Object Management Group, "Business Process Model and Notation (BPMN) Version 2.0," 03 01 2011. [Online]. Available: <http://www.omg.org/spec/BPMN/2.0/>.
- [25] Trisotech, "Resources for CMMN," [Online]. Available: <http://www.trisotech.com/tag/cmmn>. [Accessed 05 10 2017].
- [26] Object Management Group, "Case Management Model and Notation (CMMN) Version 1.1," 01 12 2016. [Online]. Available: <http://www.omg.org/spec/CMMN/1.1/>.
- [27] C. Le Clair, D. Miers, A. Cullen and J. Keenan, "Dynamic Case Management, Q1 2014," *The Forrester Wave*, 2014.
- [28] BPM Center, "BPM Center - a collaborative virtual research center in the area of BPM," [Online]. Available: <http://bpmcenter.org>. [Accessed 03 10 2017].
- [29] D. Lopresti, "Optical Character Recognition Errors and Their Effects on Natural Language Processing," *International Journal on Document Analysis and Recognition*, vol. 12, no. 3, pp. 141-151, 2009.
- [30] International Organization for Standardization, "ISO 15489-1:2016 Information and

- documentation -- Records management -- Part 1: Concepts and principles," 04 2016. [Online]. Available: <https://www.iso.org/standard/62542.html>.
- [31] EU Member States, "REGULATION (EU) No 910/2014: Electronic identification and trust services for electronic transactions in the internal market," 23 07 2014. [Online]. Available: <http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32014R0910>.
- [32] Camunda Services GmbH, "CMMN 1.1 Implementation Reference," 31 05 2017. [Online]. Available: <https://docs.camunda.org/manual/7.7/reference/cmmn11/>.
- [33] M. Kurz and C. Herrmann, "Adaptive Case Management: Supporting Knowledge Intensive Processes with IT Systems," in *S-BPM ONE '11 Proceedings of the 3rd International Conference on Subject-Oriented Business Process Management*, Ingolstadt, Germany, 2011.
- [34] M. Kurz, "Taming Diversity: A Distributed ACM-Based Approach for Cross-Enterprise Knowledge Work," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Atlanta (GA), USA, 2013.
- [35] M. Kurz and M. Lederer, "Subject-Oriented Adaptive Case Management: Extending Subject-Oriented Business Process Management to Knowledge-Intensive Cross-Enterprise Business Processes," in *S-BPM ONE '14 Proceedings of the 6th International Conference on Subject-Oriented Business Process Management*, Eichstätt, Germany, 2014.
- [36] H. R. Motahari-Nezhad, C. Bartolini, S. Graupner and S. Spence, "Adaptive Case Management in the Social Enterprise," in *Proceedings of the 10th International Conference on Service-Oriented Computing*, Shanghai, China, 2012.
- [37] M. Hauder, D. Münch, F. Michel, A. Utz and F. Matthes, "Examining Adaptive Case Management to Support Processes for Enterprise Architecture Management," in *IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, Ulm, Germany, 2014.
- [38] EU Member States, "Treaty on the Functioning of the European Union - Article 258," 25 03 1957. [Online]. Available: <http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:12008E258>.
- [39] Gartner, "Gartner IT Glossary > Identity and Access Management (IAM)," [Online]. Available: <http://blogs.gartner.com/it-glossary/identity-and-access-management-iam/>. [Accessed 05 10 2017].
- [40] F. P. Miller, A. F. Vandome and J. McBrewster, *Information Technology Audit*, London, UK: Alpha Press, 2009.
- [41] A. Schnackenberg, "Organizational Transparency: A New Perspective on Managing Trust in Organization-Stakeholder Relationships," *Journal of Management*, vol. 42, no. 7, p. 1784–1810, 2016.
- [42] D. Radigan, "Atlassian - 7 steps to a beautiful and useful agile dashboard," [Online]. Available: <https://www.atlassian.com/blog/jira-software/7-steps-to-a-beautiful-and-useful-agile-dashboard>. [Accessed 05 10 2010].

- [43] Directorate-General for Informatics (European Commission), "PM² Project Management Methodology Guide," 29 11 2016. [Online]. Available: <https://publications.europa.eu/en/publication-detail/-/publication/0e3b4e84-b6cc-11e6-9e3c-01aa75ed71a1>.
- [44] DG Taxation and Customs Union, "Customs duties mean revenue," [Online]. Available: https://ec.europa.eu/taxation_customs/facts-figures/customs-duties-mean-revenue_en. [Accessed 01 10 2017].
- [45] DG Taxation and Customs Union, "UCC BPM," [Online]. Available: https://ec.europa.eu/taxation_customs/business/union-customs-code/ucc-bpm_en. [Accessed 01 10 2017].
- [46] DG Trade, "Negotiations and agreements," [Online]. Available: <http://ec.europa.eu/trade/policy/countries-and-regions/negotiations-and-agreements>. [Accessed 01 10 2017].
- [47] Council of the European Union, "Infographic - EU trade map," [Online]. Available: <http://www.consilium.europa.eu/en/infographics/eu-trade-map-2017>. [Accessed 01 10 2017].
- [48] DG Trade, "Trade negotiations step by step," 09 2013. [Online]. Available: http://trade.ec.europa.eu/doclib/docs/2012/june/tradoc_149616.pdf.
- [49] M. A. Marin, M. Hauder and F. Matthes, "Case Management: An Evaluation of Existing Approaches for Knowledge-Intensive Processes," in *BPM 2015, 13th International Workshops*, Innsbruck, Austria, 2015.
- [50] Microsoft, ".NET," [Online]. Available: <https://www.microsoft.com/net/>. [Accessed 05 10 2017].
- [51] TIOBE, "Programming Community Index for October 2017," [Online]. Available: <https://www.tiobe.com/tiobe-index/>. [Accessed 07 10 2017].
- [52] P. Carbonnelle, "PYPL PopularitY of Programming Language," [Online]. Available: <http://pypl.github.io/PYPL.html>. [Accessed 07 10 2017].
- [53] IEEE Spectrum, "The 2017 Top Programming Languages," 18 07 2017. [Online]. Available: <https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>.
- [54] J. Gosling, B. Joy, G. Steele, G. Bracha and A. Buckley, "The Java Language Specification - Java SE 8 Edition," 13 02 2015. [Online]. Available: <https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf>.
- [55] N. Salnikov-Tarnovski, "Most Popular Java Application Servers (2017 Edition)," 30 05 2017. [Online]. Available: <https://dzone.com/articles/most-popular-java-application-servers-2017-edition>.
- [56] Apache Software Foundation, "Apache Tomcat Versions," [Online]. Available: <http://tomcat.apache.org/whichversion.html>. [Accessed 08 10 2017].

- [57] RebelLabs, "Java Tools and Technologies Landscape Report 2016," ZeroTurnaround, Tartu, EE, 2016.
- [58] Pivotal Software, "Spring Framework," [Online]. Available: <https://projects.spring.io/spring-framework/>. [Accessed 07 10 2017].
- [59] OWASP, "Top 10 2007-Broken Authentication and Session Management," [Online]. Available: https://www.owasp.org/index.php/Top_10_2007-Broken_Authentication_and_Session_Management. [Accessed 08 10 2017].
- [60] E. Gamma, J. Vlissides, R. Johnson and R. Helm, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.
- [61] I. Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, Redwood City (CA), USA: Addison-Wesley, 2004.
- [62] Pivotal Software, "Spring Expression Language (SpEL)," [Online]. Available: <https://docs.spring.io/spring/docs/3.0.x/reference/expressions.html>. [Accessed 07 10 2017].
- [63] Pivotal Software, "Web on Servlet Stack," [Online]. Available: <https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>. [Accessed 07 10 2017].
- [64] TutorialsPoint, "Spring - MVC Framework," [Online]. Available: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm. [Accessed 08 10 2017].
- [65] Pivotal Software, "Spring Web Flow," [Online]. Available: <https://projects.spring.io/spring-webflow/>. [Accessed 08 10 2017].
- [66] Pivotal Software, "Spring Security Architecture," [Online]. Available: <https://spring.io/guides/topicals/spring-security-architecture/>. [Accessed 08 10 2017].
- [67] Apache Software Foundation, "ApacheDS," [Online]. Available: <http://directory.apache.org/apacheds/>. [Accessed 07 10 2017].
- [68] J. Sermersheim, "RfC 4511: Lightweight Directory Access Protocol (LDAP)," 06 2006. [Online]. Available: <https://tools.ietf.org/rfc/rfc4511.txt>.
- [69] D. Alur, J. Crupi and D. Malks, Core J2EE Patterns - Best Practices and Design Strategies, Palo Alto (CA), USA: Sun Microsystems, 2003.
- [70] Red Hat, "Hibernate ORM," [Online]. Available: <http://hibernate.org/orm/>. [Accessed 07 10 2017].
- [71] Oracle, "JSR-000221 JDBC API Specification 4.3," [Online]. Available: <https://jcp.org/aboutJava/communityprocess/mrel/jsr221/index3.html>. [Accessed 20 10 2010].
- [72] "Hibernate ORM - Envers," [Online]. Available: <http://hibernate.org/orm/envers/>. [Accessed 07 10 2017].

- [73] Alfresco Software, "Activiti User Guide," [Online]. Available: <https://www.activiti.org/userguide>. [Accessed 12 10 2010].
- [74] World Wide Web Consortium, "Web Services Architecture," 11 02 2004. [Online]. Available: <https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>.
- [75] Alfresco Software, "Alfresco Activiti | Enterprise BPM," [Online]. Available: <https://www.alfresco.com/products/business-process-management/alfresco-activiti>. [Accessed 14 10 2017].
- [76] A. Bonham, "Comparing and Contrasting Open Source BPM Projects," 28 09 2016. [Online]. Available: <https://medium.com/capital-one-developers/comparing-and-contrasting-open-source-bpm-projects-196833f23391>.
- [77] D. Meyer, "Camunda Engine Evolution since Activiti Fork," 19 10 2016. [Online]. Available: <https://blog.camunda.org/post/2016/10/camunda-engine-since-activiti-fork/>.
- [78] J. Potts, "Activiti founders fork the project to create Flowable, an open source BPM engine," 15 10 2016. [Online]. Available: <https://ecmarchitect.com/archives/2016/10/15/4192>.
- [79] Apache Software Foundation, "Apache Tiles," [Online]. Available: <https://tiles.apache.org/>. [Accessed 14 10 2017].
- [80] Oracle, "JavaServer Pages Technology," [Online]. Available: <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>. [Accessed 14 10 2017].
- [81] Twitter, "Bootstrap," [Online]. Available: <http://getbootstrap.com/>. [Accessed 14 10 2017].
- [82] Open Group, "Service-Oriented Architecture – What Is SOA?," [Online]. Available: <http://www.opengroup.org/soa/source-book/soa/p1.htm>. [Accessed 05 10 2017].
- [83] F. Hirsch, J. Kemp and J. Ilkka, Mobile Web Services: Architecture and Implementation, Chichester, UK: John Wiley & Sons, 2007.
- [84] Apache Software Foundation, "Apache Chemistry - What is CMIS?," [Online]. Available: <http://chemistry.apache.org/project/cmisis.html>. [Accessed 05 10 2017].
- [85] Camunda Services GmbH, "BPMN Workflow Engine," [Online]. Available: <https://camunda.org/>. [Accessed 22 10 2017].
- [86] Flowable, "Java Business Process Engines," [Online]. Available: <http://www.flowable.org/>. [Accessed 22 10 2017].
- [87] Object Management Group, "BPMN, CMMN and DMN Specifications at OMG," [Online]. Available: <http://www.omg.org/intro/TripleCrown.pdf>. [Accessed 22 10 2017].
- [88] Trisotech, "Triple Crown of Process Improvement Standards," [Online]. Available: <https://www.trisotech.com/infographics/bpmn-cmmn-dmn-poster>. [Accessed 22 10 2017].