Case Management Model and Notation

Analyse van een toepassing in de praktijk
Titel: Case Management Model and Notation, analyse van een toepassing in de praktijk
Title: Case Management Model and Notation, analysis of a practical application

Opleiding:
Open Universiteit, faculteit Management, Science & Technology
Masteropleiding Business Process Management and IT

Naam afstudeerder: Pieter Jansen
Studentnummer: 83.19.19.859

Datum presentatie: 15 december 2015

Afstudeercommissie:
1e begeleider: Dr. Ir. E. Roubtsova
2e begeleider: Dr. A. Counotte
Examinator: Prof. Dr. Ir. S. Joosten

cursuscode: T9232B
Abstract

The development and implementation of software to support case management should be based on standards that fit into a process that supports continuous change in the business environment.

The Case Management Model and Notation (CMMN) specification published by the OMG®, which defines a common meta model for modelling and graphically expressing a case and an interface format for exchange of case models amongst different tools, should fit into this category of standards.

Although the standard contains a lot of wisdom its application, operation and best practices are not trivial for many organisations. The main question is if CMMN can be applied in practice. This thesis gives an answer to this question.

As part of a literature study the theoretical background and context of CMMN are defined.

Reference is made to business artifact development and case management and the common characteristics of both. The specific design, planning and run-time execution concept of CMMN is explained.

Starting from a global CMMN meta model, the semantics of the CMMN development elements and run-time execution are explained by using sample implementations. A short description of the CMMN notation is given.

It is explained how other development methods which are related to business artifact development for example focusing on GSM (Guard-Stage-Milestone) can partially be used for the development of a CMMN model.

A hypothetical business case reflecting how a patent organisation would apply the CMMN standard is explained in order to test the application of the CMMN standard. This was specifically focussing on the design aspect of the standard. The business case is based on a part of the search process, which takes place after a patent application has been filed.

Using a top-down development approach, including the definition of scenario’s, a CMMN model is created for 2 sub-processes.

Experimental CMMN tools are used for creating the design.

Based on observations and conclusions, recommendations are made as to how the CMMN standard could be improved.
# Table of Contents

Abstract 4
Table of Contents 5
List of Figures 7
List of Tables 7
List of Appendix Figures 8
1. Introduction 9
1.1. Background 9
1.2. CMMN 9
1.3. Report structure 9
1.4. Expected audience who can benefit 9
2. Research framework 11
2.1. Problem 11
2.2. Research questions 11
2.3. Method 11
2.3.1. Research model 11
2.3.2. Methods and steps 12
3. Theoretical framework 14
3.1. Theoretical evolution from business artifact to CMMN 14
3.1.1. Business artifacts, case and case management 14
3.2. CMMN theoretical conceptual meta model 17
3.2.1. Introduction 17
3.2.2. Design-time modelling and Run-time planning 17
3.2.3. CMMN semantics and notation 20
3.2.4. CMMN Interaction scheme and execution semantics 29
3.3. Development methods to get a CMMN specification 34
3.3.1. Introduction 34
3.3.2. An artifact centric design methodology 34
3.3.3. Synthesizing a GSM model from a Business Process Model 35
4. Experiment at the sample patent office 37
4.1. Introduction 37
4.2. CMMN key concepts 37
4.3. Development approach 38
4.4. Experiment 38
4.4.1. Introduction 38
4.4.2. Experimental steps 38
4.4.3. Division allocation and Search validation 41
4.5. Observations 46
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Conclusions and recommendations</td>
<td>48</td>
</tr>
<tr>
<td>References</td>
<td>51</td>
</tr>
<tr>
<td>Appendix A. Sample patent office</td>
<td>53</td>
</tr>
<tr>
<td>Appendix B. GSM (Guard Stage Milestone) semantics</td>
<td>55</td>
</tr>
<tr>
<td>Appendix C. CMMN B-step</td>
<td>61</td>
</tr>
<tr>
<td>Appendix D. CMMN form</td>
<td>64</td>
</tr>
<tr>
<td>Appendix E. Process model search process</td>
<td>88</td>
</tr>
<tr>
<td>Appendix F. YAWL notation</td>
<td>92</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2-1 Research model ........................................................................................................ 12
Figure 3-1 Design-time modelling versus Run-time planning .................................................... 19
Figure 3-2 CMMN model in XML notation [15] ....................................................................... 20
Figure 3-3 CMMN Notation ...................................................................................................... 28
Figure 3-4 CMMN lifecycle concepts ....................................................................................... 30
Figure 3-5 Example to explain the CMMN lifecycle concept ................................................... 31
Figure 4-1 Part of the sample patent office's search process ..................................................... 39
Figure 4-2 Example of CMMN Information model elements .................................................... 40
Figure 4-3 Process flow application from filing until allocation ............................................... 42
Figure 4-4 Stage Division Allocation ....................................................................................... 43
Figure 4-5 Search validation of a patent application ................................................................. 44
Figure 4-6 Stage Search validation ............................................................................................ 45
Figure 4-7 Stage Clarification .................................................................................................... 45

List of Tables

Table 3-1 Behaviour property rules ......................................................................................... 26
Table 5-2 Conclusions and recommendations ......................................................................... 48
# List of Appendix Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-1</td>
<td>Semantic GSM Artifact-centric model</td>
<td>55</td>
</tr>
<tr>
<td>B-2</td>
<td>Example GSM implementation</td>
<td>56</td>
</tr>
<tr>
<td>C-1</td>
<td>Interaction scheme for B-step</td>
<td>62</td>
</tr>
<tr>
<td>C-2</td>
<td>Sample CMMN interaction scheme</td>
<td>63</td>
</tr>
<tr>
<td>E-1</td>
<td>Process model search process part 1</td>
<td>89</td>
</tr>
<tr>
<td>E-2</td>
<td>Process model search process part 2</td>
<td>89</td>
</tr>
<tr>
<td>E-3</td>
<td>Process model search process part 3</td>
<td>90</td>
</tr>
<tr>
<td>E-4</td>
<td>Roles executing the activities</td>
<td>90</td>
</tr>
<tr>
<td>F-1</td>
<td>YAWL process modelling notation</td>
<td>92</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Background

This thesis presents the results of research on the theoretical concepts and the practical application of CMMN modelling in the domain of patent application management.

The sample patent office, referred to in the document, provides a uniform application procedure to enable inventors to seek patent protection. A description of the sample patent office can be found in Appendix A.

1.2. CMMN

In 2014 the OMG® (Object Management Group) has published a specification named CMMN (Case Management Model and Notation) for modelling and graphically expressing a case in respect to case management.

CMMN was designed as a declarative approach for the design, run-time planning and execution of business cases.

CMMN defines a common meta model and notation to graphically express a (business) case and its ‘business artifacts’ with respect to case management.

1.3. Report structure

The report structure of this thesis is as follows.

Chapter 1 ‘Introduction’ provides an introduction to the thesis. It provides information about the report structure and defines the audience who can benefit from this thesis.

Chapter 2 ‘Research framework’ contains information about the formulation of the central research questions and sub questions and the organisation of the research by means of a research model.

Chapter 3 ‘Theoretical framework’ provides information about the theoretical background and context of CMMN as a result of an extensive literature study.

Chapter 4 ‘Experiment at the sample patent office’ explains the development approach, design deliverables and observations as a result to apply and test the concepts of the CMMN standard proposal in practice.

Chapter 5 ‘Conclusions and recommendations’ contains conclusions and recommendations as a result of discussion and answers the central research question.

Chapter 6 ‘Research Process Reflection’ reflects the research.

Finally, appendices providing supporting information, are included at the end of the document.

1.4. Expected audience who can benefit

The information in this thesis can be useful for:
- teachers and students who are teaching or studying in the field of declarative business process modelling and case management,
- organisations who are considering the purchase or usage of case management software products and need to define the requirements for selection,
- persons who are involved in design, development and implementation of case management software systems,
- organisations who are considering the use of software products which support the CMMN standard proposal.
2. Research framework

2.1. Problem
The problem for this research comes from the appearance of the CMMN standard proposal. Although this standard contains a lot of wisdom its application, operation and recognition are not trivial for organisations.

2.2. Research questions
In order to approach the problem, the following research questions have been defined:

Central research question:
Can CMMN be applied in practice?

To answer this central research question, the following six sub-questions have been defined:

1. What is the theoretical background and context of CMMN?
2. What is the theoretical conceptual meta model of CMMN?
3. What are the development methods to get a CMMN specification?
4. Which CMMN key concepts are relevant to test its practical application?
5. What are the key concepts found in a typical elicitation project which follows a model-driven development approach for a case management system and how can these concepts be used as test scenarios to test the applicability of CMMN?
6. What conclusions can be made and recommendations given concerning the application, operation and best practices of CMMN in practice?

2.3. Method

2.3.1. Research model
For this research the following research model has been applied (see Figure 2-1 Research model).
2.3.2. Methods and steps

The methods applied are presented as ‘a’, ‘b’ and ‘c’ between the arrows at the bottom in Figure 2-1.

These methods are:

a. Literature study,
b. Experiment,
c. Discussion, comparison of the literature study and the experiment results.

The methods have been used in the following phases and steps to answer the research questions in subsequent chapters of this document:

1) As part of a literature study (a), and using theory on CMMN, the theoretical background and context of CMMN will be explained, a CMMN conceptual model will derived from the UML-GSM definition and the CMMN notation delivering a theoretical conceptual (meta) model. Development method(s) for artifact discovery and data-centric modelling will be listed (if necessary combined) and selected for the empirical study.

This step will answer the research questions 1 (RQ1), 2 (RQ2) and 3 (RQ3).

2) During the empirical study an experiment (b) has been performed to test the practical application of CMMN concepts against a practical case. First relevant CMMN concepts for the test have been defined. Thereafter a part of the hypothetical patent office’s requirements related to its search process have been considered for the definition of test case samples in terms of scenarios. The test case samples have been based on a subset of the requirements and key concepts formulated. In respect to the CMMN elements, the scope of the experiment is restricted to design and does not include any authorisation and authentication aspects. The experiment has delivered a specification including a practical behaviour model of this process.

This step will answer research question 4 (RQ4) and 5 (RQ5).
3) By means of a discussion (c) a comparison will take place between the theoretical conceptual (meta) model and the practical declarative conceptual model in order to check the coverage of concepts. Concepts useful and/or missing in the theoretical conceptual (meta) model will be identified and recommended.

In this step research question 6 (RQ6) will be answered.
3. Theoretical framework

In each of the following sections an answer has been given to the research questions 1, 2 and 3 following an extensive literature review.

For this review the Digital Library of the Dutch Open University has been used, in particular the information files and search engines related to Informatics and Management Sciences. For the search there was no restriction on date published.

The following list provides an overview of search terms, which have been used (in combination) for a first inventory:

- Business artifact,
- Case,
- Case management,
- GSM,
- CMMN.

3.1. Theoretical evolution from business artifact to CMMN

3.1.1. Business artifacts, case and case management

This section provides an answer to:

**Research question 1: what is the theoretical background and context of CMMN?**

**Business artefact.** This term is used to represent a product or service [6]. Many business products and services are complex entities.

An example of a business artifact could be a purchase order with specific attributes like vendor ID, product type and price.

A ‘business artifact’, a.k.a. business entity (with lifecycle), is a key conceptual entity that is central to guiding the operations of a business, and whose content changes as they move through those operations [9].

An artifact type includes two models:

- an *Information Model* that captures all of the business-relevant data about entities of that type in attribute/value pairs in either materialized or virtual form, and
- a *Lifecycle model*, that specifies the possible ways an entity of that type might progress through the business from its creation until its final form, and the ways in between how it will respond to events and invoke external services, including human activities.

An artefact instance is an instantiation of the corresponding information and lifecycle models of the artifact type.

Business artifacts evolve during the business processes. This is described by the business artifacts lifecycles in terms of its business relevant stages. Changes in these stages are performed by tasks (services) and thus require an update of the business artifact.

**Case.** There is a strong relationship between the ‘business artifact’ paradigm and a ‘case’. A case is
an instance of a business artifact.

A case is a powerful abstraction applied to a subject. The subject of a case may be: a person, a legal action, a business transaction, or some other focal point around which actions are taken to achieve an objective [6].

A case is a proceeding or process instance that involves actions taken regarding a subject in a particular situation to achieve a desired outcome. E.g. a legal case involves the application of the law to a subject in a certain fact situation. A case can change in an unpredictable, dynamic and ad hoc way as it is progressed through an organization.

Case management is an approach of ‘coordination of services on behalf of an individual person (= client)’ [3]. It entails collection of diverse data including documents, records, e-mails etc. They can be in different formats, structured and unstructured and can come from different sources. There is a need to effectively store, maintain and retrieve case information. Case data should be always provided in the context of current state of the case.

Software products and procedures for case management have emerged to assist the different participants to make decisions or perform certain tasks related to cases. Originally case management was developed to help manage social work and related application areas. It is strongly based on data as the typical product of these processes.

Dynamic Case Management (DCM), sometimes called ‘Adaptive Case Management’ (ACM), is both a human- and a technology-based approach that is driven by incoming events [4]. In the context of this study the definition for Dynamic Case Management is taken [5].

“A highly structured, but also collaborative, dynamic, and information-intensive process that is driven by outside events and requires incremental and progressive responses from the business domain handling the case.

Examples of case folders include a patient record, a lawsuit, an insurance claim, or a contract, and the case folder would include all the documents, data, collaboration artifacts, policies, rules, analytics, and other information needed to process and manage the case.” A patent handling record that will be studied in this thesis is also an example of a case folder.

Case management planning is concerned with determination of which tasks are applicable, or which follow-up tasks are required, given the state of the case. Decisions may be triggered by events or new facts that continuously emerge during the course of the case, such as the receipt of new documents, completion of certain tasks, or achieving certain milestones. Individual tasks that are planned and executed in the context of the case might be predefined procedural processes in themselves, but the overall case cannot be orchestrated by a predefined sequence of tasks [6].

A case file is a collection of data about the case. Modelling of constraints and guidance on the actions to be taken in a case requires the specification of rules that reference the data in the case file. Cases are directed not just by explicit knowledge about the particular case and its context represented in the case file, but also by explicit knowledge encoded as rules by business analysts, the tacit knowledge of human participants, and tacit knowledge from the organization or community in which participants are members [6].

Within the artifact paradigm and case management there is a strong emphasis on conceptual entities that evolve as they move through a business. Both approaches support an ad hoc style of managing activities and make data a first-class citizen. In particular, both call for maintaining an integrated view of all data that is business-relevant to a given entity (case) instance as it evolves.

Related standards for Business Process Management.

Recent BPM (Business Process Management) paradigms organise processes around business artifacts and cases [7].
De Man [7] found that procedurally based business artifacts provided useful aspects applicable to case management, but not enough to cover all uses.

One of the approaches to support case management using business artifacts is called GSM (Guard-Stage-Milestone). The GSM approach for business artifacts was developed to provide a hierarchical, modular, and more declarative approach for specifying artifact lifecycles. GSM is standardized in UML (Unified Modelling Language) and provides a formal foundation. It is a declarative approach and not procedural as, for example Petri Nets. Would the business accept this approach of modelling? It depends on how easy and intuitive the modelling of business processes in GSM can take place and how precise the models of business processes in GSM can be defined. Key foundational elements of CMMN are derived from the GSM.

The GSM meta model is a formalism for describing artifact-based processes, called Guard-Stage-Milestones [9] [10].

Guard-Stage-Milestones are variants of the Event-Condition-Action rules paradigm, and centred around a GSM business step (also called GSM B-step), which focuses on the full impact of incorporating the Incoming Event.

The GSM meta model reflects artifacts, or artefact types, being key business entities of a given domain, which are characterised by:

1. an Information model that captures the data maintained by an artifact. This model consists of data attributes which represent data relevant to the business, status attributes which hold information about the progress of the artifact instance along its lifecycle and last event type information,
2. a Lifecycle model that specifies the possible progressions of an artifact, and how the underlying data are manipulated as the result of a progression step,
3. a set of artifact instances, which are instantiations of the corresponding information and lifecycle models of an artifact type. The description of a particular business process may involve several instances of different artifact types.

The original semantics of GSM by means of a GSM meta model is most completely defined in [12] and [13]. A simplification of the semantics is defined in [13].

The semantics of GSM is briefly described in Appendix B ‘GSM (Guard Stage Milestone) semantics’. In [10], [2], [12] and [13] examples of GSM models are presented and explained.

Building on the application of GSM for description of artefact-based processes, the CMMN standard specification is intended to capture the common elements (stages, tasks, conditions), that can be used to build the case management software products.

The CMMN specification describes:

- the semantics of the common elements of the case management model,
- the execution semantics described by the lifecycle of instances within the context of the case management model,
- a format for exchanging case models among different tools.

It is important to remember that CMMN is a standard and GSM is more an approach to support artifact-centric development which is not formalised.

Although the purpose of this research is not to analyse the differences between CMMN and GSM it can be stated that CMMN is influenced by GSM, e.g. it also contains stages, tasks, milestones and sentries, it is not an extension to GSM as there are fundamental differences between both e.g.:
- with respect to design in GSM, for a single artifact and a single stage S, that stage S can have at most one occurrence that is executing at one time. In contrast, in CMMN, for a single case instance, a given stage S’ might have multiple occurrences executing at the same time.

- Condition language: GSM includes a rich condition language, based on OCL that can also specify conditions that span across multiple artifact instances. In contrast, in CMMN, conditions (e.g. for guards and milestones) can refer only to attributes within a single case instance which does appear to be restrictive.

- CMMN supports so-called design-time modelling and run-time planning and execution concepts which are not supported by GSM.

3.2. CMMN theoretical conceptual meta model

3.2.1. Introduction

This section provides an answer to:

**Research question 2: what is the theoretical conceptual meta model of CMMN?**

In 3.2.2. ‘Design-time modelling and Run-time planning’ the design-time and run-time phase characteristics of CMMN are explained.

Subsequently, in 3.2.3. ‘CMMN semantics and notation’ the CMMN conceptual meta model will be presented and explained. The purpose of this section is to provide a global overview of the semantics and notation of CMMN.

Finally, in 3.2.4. ‘CMMN Interaction scheme and execution semantics’ the CMMN interaction scheme and execution semantics are explained based on the life cycle of different CMMN components. The CMMN behaviour is visualised based on an example.

3.2.2. Design-time modelling and Run-time planning

Within the context of CMMN, a case has two distinct phases (see Figure 3-1 Design-time modelling versus Run-time planning):

- The **design-time phase**:

  during this phase, business analysts are engaged in modelling, which includes defining stages or tasks that are:
  1. required and always part of pre-defined segments in the case model,
  2. optional so not required to complete for the scope to terminate and
  3. discretionary implying, they can be selected (instantiated) by case workers at run-time under condition of parameters specified in planning tables.

  A process model specified at design time serves as a case management plan for execution of a case.

- The **run-time planning and execution phase**:

  during this phase, case workers create and execute the plan, particularly by performing tasks as planned, while the plan may continuously evolve, due to the same or other case workers being engaged in planning and execution, i.e. adding so-called ‘discretionary plan-items’ (tasks and stages) to the plan of the case instance at run-time. This fundamental characteristic of case management, supported by CMMN (but not by GSM), is referred to as
‘runtime flexibility’ [18]. Runtime flexibility allows process participants to respond to challenges or new requirements that were not considered during planning the business processes. This run-time planning is based on information that has become available to the case. Changing the case behaviour means changing a case process. This includes changing the processing of running cases.
Figure 3-1 Design-time modelling versus Run-time planning
3.2.3. CMMN semantics and notation

3.2.3.1. CMMN meta model

A global graphical presentation of the CMMN semantical model in XML notation is provided in Figure 3-2 CMMN model in XML notation [15]. This model is an elaboration based on [8] defined by [15]. The meta model encompasses both the design-time model and run-time planning model.

An arrow presents a specialisation and a diamond presents a composition relationship between two classes.

*CMMNEl**ement* is the abstract base class for all other classes in the case meta model.

*Case* is a top-level concept that combines all elements that constitute a case model.

A *case* instance is composed of:
1. *Information Model* elements in a *caseFileModel*,
2. *Behaviour or plan model elements* in a *casePlanModel*,
3. *Roles* that correspond to humans expected to participate in the case in *caseRoles*,
4. *Inputs and outputs* to enable interaction of the case with its environment.

When a case instance is created, the caseFileModel, casePlanModel, and caseRoles are all initialised. The stage instance implementing the casePlanModel starts executing in an active state.

A case instance has its own lifecycle.

![Figure 3-2 CMMN model in XML notation [15]](image-url)
3.2.3.2. CMMN component semantics

The semantics of the CMMN components are described in the following sections for each case management model component as presented in Figure 3-2 or [6].

3.2.3.2.1. caseFileModel

The Information model of a case, which is represented by a caseFileModel and not part of the standard, comprises of set of classes for the management of the information (data) aspects of a case. This model supports, amongst other, in particular the information structure of the CMIS (Content Management Interoperability Services) standard for content management systems.

Important elements of the caseFileModel are caseFile and caseFileItem.

A caseFile consists of caseFileItem(s) which may represent a piece of information of any nature, ranging from unstructured to structured, and from simple to complex, which information can be defined based on any information modelling “language”. The structure, as well as the ‘language’ (or format) to define the structure, is defined by an associated CaseFileItemDefinition (not presented in Figure 3-2, see [6]).

A caseFile is some kind of logical information model. Information in the case files serves as a context for raising events and evaluating expressions as well as reference point for caseParameters, i.e. inputs and outputs of tasks.

A caseFile also serves as a container for data that is accessible (through caseParameters) to other systems and people outside the case.

A caseFileItem can also be a piece of information of any nature ranging from unstructured to structured, and from simple to complex as part of a caseFile for example a folder, an entire folder hierarchy referring or containing other caseFileItems or simply an XML document.

3.2.3.2.2. casePlanModel

For a particular case model, a casePlanModel comprises:
- all elements that represent the initial plan of the case such as stages, tasks and milestones, and
- all elements that support the further evolution of the plan through run-time planning by case workers.

The execution semantics (behaviour) is in particular described by the lifecycle of some important object instances as part of:
- the casePlanModel: in this context the 4 PlanItemDefinition elements are EventListener, PlanFragment (stage), task and milestone and
- the before mentioned caseFileModel: object caseFileItem.

A class PlanItemControl (not presented in Figure 3-2, see [6]) specifies defaults for aspects of control for before mentioned casePlanModel elements, such as whether these instances have to be completed and/or can start according to defined principles.

The semantics of each PlanItemDefinition object is described below.

3.2.3.2.2.1. EventListener

CMMN attempts to predefine events and their respective causes. These predefined CMMN events are referred to as standard events and can be categorised into two areas:
- anything that can happen to information that denotes a transition in a defined lifecycle of CaseFileItems in the CaseFile.

- anything that happens to stages, tasks and milestones that denote transitions in the defined lifecycle of these.

The elapse of time cannot be captured via these above categorised standard events and for this reason, the class EventListener has been introduced into the CMMN model this is an abstract class which is specialised into two concrete classes:

- TimerEventListener which is used to catch instances of PlanItemDefinition with predefined elapses of time and
- UserEventListener in order to catch events that are raised by a user and which are used to influence the proceedings of the case directly instead of indirectly, via impacting information in the caseFile.

These EventListeners have their own CMMN-predefined lifecycle, so that any elapsed time as well as any user event, can still be captured as ‘standard events’ and therefore handled in a uniform way, denoting transitions in the CMMN-defined lifecycle of the EventListener.

### 3.2.3.2.2. Milestone

A milestone is a plan element, which represents an achievable target, defined to enable evaluation of progress of the case. No work is directly associated with a milestone, but completion of a set of tasks or the availability of key deliverables (information in the caseFile) typically leads to achieving a milestone. It is not associated with any work but rather marks that certain conditions have been reached within the case. Therefore, it may have zero or more entry criteria, which define, when a milestone is reached.

As a milestone is a regular PlanItemDefinition, a milestone’s completion may be used as entry criteria for other tasks and stages. This way, a milestone can be used to bring logical stages within a case into order.

A milestone can be:
- attached to a stage to represent the goal reached when the stage has completed, or
- standalone to represent an important business condition that is not the direct result of any stage.

A milestone has its own lifecycle.

### 3.2.3.2.2.3. PlanFragment

A PlanFragment is a set (container) of PlanItems, possibly dependent on each other and optionally sentries that define the criteria according to which the PlanItems are enabled (or entered) and terminated (or exited). PlanFragments can represent PlanItem-and-Sentry patterns of any complexity.

Examples of PlanFragments are:
- a combination of two tasks, whereby, the completion of one task satisfies the sentry that enables the start of the other (sequence flow),
- a combination of an event listener and a task, whereby the occurrence of the event satisfies the sentry that enables the start of the task.

PlanFragments being stages have lifecycles. PlanFragments, not being stages do not have lifecycles.
### 3.2.3.2.4. PlanItem and PlanItemDefinition

A PlanItem object is a use of a PlanItemDefinition element in a PlanFragment (or stage). CMMN differentiates between PlanItems and PlanItemDefinitions. While PlanItems represent actual units of work that are enacted as part of the case, PlanItemDefinitions serve as the blueprint for how a PlanItem has to be enacted. This concept simplifies the reuse of PlanItemDefinitions and furthermore enables dynamic planning such that additional items can be generated at runtime from a definition.

Apart from reuse of configuration, PlanItemDefinitions can be instantiated at runtime, typically referred to as planning. Planning allows users to create plan items of well-defined set of PlanItemDefinitions dynamically as needed.

Transitions in a CMMN Case can happen in two ways:
- either by external interaction or
- by events occurring and conditions being fulfilled.

The former refers to any explicit interaction with a case that is triggered from outside. For example, a caseworker completing a human task would be such an interaction. Completing a task means that the corresponding PlanItem is completed, depending on the actual case model, the case instance may be complete.

Similar changes in the state of a case may be driven by events occurring or conditions getting fulfilled. For example, it is possible to define that when one PlanItem completes, another is enabled. Similarly, a PlanItem can terminate when an event triggers. When specifying PlanItems, this concept is referred to as *entry criteria* and *exit criteria*. These criteria are always defined for individual PlanItems, not for PlanItemDefinitions. The conditions and events behind entry and exit criteria can be expressed by so-called sentries.

When any entry criterion is met, the PlanItem performs a state transition from ‘available’ to ‘enabled’.

Similarly, when any exit criterion is met, a PlanItem performs a state transition from any of the states ‘available’, ‘disabled’ or ‘active’ to ‘terminated’.

A Milestone should have at least one entry criterion. It does not have exit criteria.

The details of the PlanItem states and transitions are provided in 3.2.4. ‘CMMN Interaction scheme and execution semantics’.

### 3.2.3.2.5. Sentry

PlanItems may have associated Sentries.

A sentry is a criterion to enable or terminate a task or stage or to achieve or invalidate a milestone. It 'watches out' for important situations to occur (or events), which influence the further proceedings of a case. It is a combination of an event and/or a condition.

A sentry may consist of two parts:
- zero or more OnParts that specify the event(s) that serves as a trigger,
- zero or more IfParts that specify conditions.

A sentry MUST have an IfPart or at least one OnPart.

An IfPart defines an additional condition that is checked when all OnParts of the sentry are fulfilled.

When the event is received, the condition might be applied to evaluate whether the event has effect or not. If all On-Parts of a sentry have occurred, and its If-Part (if existent) evaluates to ‘true’, the sentry is ‘satisfied’.

A sentry that is satisfied triggers the PlanItem that refers to it in the PlanFragment:
- when the sentry is referenced by one of the PlanItem’s entry criteria: a task or stage will be enabled, and a milestone will be achieved.
- when the sentry is referenced by one of the PlanItem’s exit criteria: a task or stage will be terminated (exited).

A sentry and the task correspond to an ECA (Event-Condition-Action) rule. An ECA rule, is not a business rule, and represents rules for process/case execution on the application layer.

Sentries may take one of the following forms:

1. An event part and a condition part in the form: on <event> if <condition>
2. An event part in the form: on <event>
3. Just a condition part in the form: if <condition>

Sentries allow a flexible definition of event occurrences and data-based conditions to be fulfilled. The following rules apply for combining OnParts and IfParts:
- a valid sentry must have at least one OnPart or an IfPart,
- a sentry without OnParts is fulfilled when the IfPart evaluates to ‘true’,
- a sentry without an IfPart is fulfilled when all OnParts have occurred.

A rule is a sentry along with an internal action, namely opening/closing (terminating) a stage or achieving/invalidating a milestone.

### 3.2.3.2.6. Stage

A stage is a cluster of business-relevant activity that might be performed for, with, and/or by a case instance, in order to achieve one (or more) of the milestones owned by that stage.

A stage can contain any element required to construct and further evolve case plans.

A stage can contain other PlanItems like tasks and other stages. It is a recursive concept — stages can be nested within other stages. The ‘most outer’ stage to the case is its casePlanModel.

A stage has its run-time representation, thus its progress and completion can be tracked based on its lifecycle.

### 3.2.3.2.7. Task

A task is an atomic unit of business relevant work that is to be performed by an outside agent (human or machine).

There are three types of tasks:
- **human task**: a task that is performed by a case worker, they can be:
  - Blocking: task is waiting until the work associated with the task is completed,
  - Non-Blocking: the task is not waiting for the work to complete and completes immediately, upon instantiation.
- **process task**: can be used in the case to call (invoke) a business process. Also a process task can be ‘Blocking’ or ‘Non-Blocking’. It is possible to exchange variables between the process task (in a case instance) and the process task that it creates.
- **case task**: can be used to call another case(task). Also a case task can be ‘Blocking’ or ‘Non-Blocking’. It is possible to exchange variables between the case task (in a case instance) and the case task it creates.

Given that an entry criterion is fulfilled, there are two ways to activate a task:
- by manual activation of a user,
- by automatic activation.

By specifying a manual activation rule, a user can decide to activate a task or instead disable it.

Any task type can be discretionary.

A task has its its own lifecycle model.

### 3.2.3.2.8 Planning table

Planning is a run-time effort. A planning table (not presented in Figure 3-2, see [6]) defines the scope of planning, in terms of identifying a sub-set of PlanItemDefinitions that can be considered for planning in a certain context. A planning table can have several table items (i.e. Planning tables and discretionary items not presented in Figure 3-2, see [6]).

Planning tables can be assigned to:
- a stage: the planning table can be used to plan instances of tasks and stages into that stage instance.
- a human task: the planning table can be used to plan instances of tasks and stages into the stage that contains the human task.

With planning tables, it is possible to make discretionary items dynamically applicable for planning using applicability rules. These rules specify, whether a table item is "applicable" ("eligible", "available") for planning, based on conditions that are evaluated over information in the caseFile.

### 3.2.3.2.9 Behaviour property rules

Behaviour property rules are used to derive Boolean values that can influence the execution of a Case instance. See Table 3-1 Behaviour property rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManualActivationRule</td>
<td>Determines whether the task or stage instance should move to state 'enabled' or 'active'. This rule is evaluated and used when the entry criteria of the task or stage instance is satisfied. If this rule evaluates to &quot;true&quot;, the stage or task instance transitions from 'available' to 'enabled' (and it must wait for manual activation), otherwise it transitions from 'available' to 'active'.</td>
</tr>
<tr>
<td>RequiredRule</td>
<td>Determines whether the milestone, stage or task having this condition must be in the 'completed', 'terminated', 'failed' or 'disabled' state in order for its parent stage instance to transition into the 'completed' state.</td>
</tr>
<tr>
<td>RepetitionRule</td>
<td>Must be evaluated when the milestone, stage or task instance is instantiated and transitions to the 'available' state, and their Boolean value should be maintained for the rest of the life of the milestone, stage or task instance.</td>
</tr>
<tr>
<td>ApplicabilityRule</td>
<td>An applicability rule, is not a business rule, and represents a rule for process/case execution on the application layer. They are used to specify, whether a table item is 'applicable' ('eligible', 'available') for planning, based on conditions that are evaluated over information in the caseFile. If the condition of the applicability rule evaluates to &quot;true&quot;, then the tableItem (Planning table and/or Discretionary item) is applicable for planning. This rule is a 'constraining' rule, as by default a table item is applicable for execution (the rule evaluates to 'true').</td>
</tr>
</tbody>
</table>
### Table 3-1 Behaviour property rules

#### 3.2.3.2.3. caseRoles

Roles are implemented by means of `caseRoles`, which authorise caseworkers or teams of caseworkers to perform human tasks, plan based activities and raise user events. Assignment of roles to participants, such as individuals or teams, is not included in the scope of CMMN.

#### 3.2.3.2.4. caseParameters

`caseParameters` model the inputs and outputs of cases and tasks.

#### 3.2.3.3. CMMN notation

##### 3.2.3.3.1. Notation elements

The CMMN specification [6] also describes the CMMN notation. The CMMN notation provides a depiction of the behavioural elements of a Case as defined in the casePlanModel (see Figure 3-3 CMMN Notation).

**CasePlan**

The `casePlan(Model)` is depicted using a folder shape with the name of the case enclosed into the upper left rectangle. All elements of a casePlanModel are located within boundaries of a case plan shape. Because of the declarative nature of CMMN relative positions of shapes have no meaning.

Every type of case plan item has an individual shape assigned to it.

**CaseFileItem**

A CaseFileItem is depicted by a “Document” shape that consist of a rectangle with a broken upper right corner.

**Stage**

A stage is depicted by a rectangle shape with angled corners. As a stage can be a composite element, they can take a collapsed or expanded form.
**Task**

A task is depicted by a rectangle with rounded corners. Depending on type and/or parameters, the task shape can have some additional symbol placed in the upper left corner. A discretionary task is depicted by a rectangle shape with dashed lines and rounded corners. If a task is blocking depends on the type of task and presented symbol. A ‘user’ symbol indicates the task is blocked and the ‘hand’ symbol indicates the task is not blocked.

**Sentry**

A sentry can be used as entry (shallow diamond shape) or exit criterion (solid diamond shape). A sentry itself has no graphical representation.

**EventListener**

An event listener is depicted by a double line circle shape.
Connectors can be used to visualise dependencies between plan items. The shape of the connector object is a dotted line.

One such depicted dependency is the On-Part of a sentry. For example, task C can be activated only:
1. If task A is complete,
2. If task A and task B are complete,
3. If task A or task B are complete.

The other type of dependency is between a human task and discretionary items in its planning table.
Planning table

A planning table is depicted by a table shape. The maker at the centre bottom cell of the planning table indicates if the discretionary items are visualised (+) or not (-).

Milestone

A milestone is depicted by a rectangle shape with half-rounded ends.

Activation

A manual activation rule is depicted by a white right arrow in the task. When no manual activation symbol is presented it is assumed ‘Manual activation’ is applicable (=true).

RequiredRule

A required rule is depicted by an exclamation mark.

RepetitionRule

A repetition rule is depicted by a hash.

3.2.4. CMMN Interaction scheme and execution semantics

The CMMN interaction scheme is, like GSM (see Appendix B), based on so-called B-steps.

As an example, a CMMN B-step is presented in Appendix C ‘CMMN B-step’.

The execution semantics of CMMN is described by the lifecycle of instances within the context of the case management model.

CMMN Case instances and PlanItems go through a lifecycle of states during their execution. Depending on their state, different actions may be carried out to interact with them. Moreover, state transitions may automatically trigger changes in other PlanItems. The concrete lifecycle of a PlanItem depends on its PlanItemDefinition.

Figure 3-4 CMMN lifecycle concepts provides an overview of the lifecycle concepts for the artifacts:

- case instance,
- task,
- stage (similar lifecycle concept as task),
- event-listener and,
- milestone (similar lifecycle concept as event-listener).
Figure 3-4 CMMN lifecycle concepts
To understand the lifecycle concept for a CMMN artifact during CMMN execution, the following case will serve as an example.

![Figure 3-5 Example to explain the CMMN lifecycle concept](image)

The case plan model Treat Fracture in Figure 3-5 Example to explain the CMMN lifecycle concept contains two human tasks (presented as 2 rounded boxes):
- task A ‘Examine Patient’ and
- task B ‘Perform XRay’.

Both tasks are connected by a connector (presented as a dotted line) and an entry criterion (sentry) presented as a diamond. When a task has no entry criterion it progresses automatically from state AVAILABLE to ENABLED.

The sentry in task B expresses that task B ‘Perform XRay’ can be enacted when task A ‘Examine Patient’ has finished.

The following activities take place for each object. To show the transitions for the life cycle of each object, a step-by-step presentation is given for each object:

1. A doctor instantiates the case. A new case instance ‘Treat Fracture’ is created in state ACTIVE.

   ![Case instance Treat Fracture ACTIVE](image)

2. Two instances for each human task are automatically created both in state AVAILABLE.

   ![Case instance Treat Fracture ACTIVE](image)
   ![Task A Examine Patient AVAILABLE](image)
   ![Task B Perform XRay AVAILABLE](image)
3. Task A has no sentry to start, so it immediately reaches state ENABLED

4. A doctor starts task A which reaches state ACTIVE. Task A is added to the doctors task list.

5. The assigned doctor completes task A. Task A reaches the state COMPLETED.

6. The transition to state COMPLETED of task A triggers the sentry of task B. Task B becomes ENABLED.
7. A radiologist starts task B which reaches state ACTIVE. Task B is assigned to the radiologists task list.

8. The assigned radiologist completes task B. Task B reaches the state COMPLETED.

9. With both tasks in state COMPLETED, in the example the case instance automatically reaches the state COMPLETED as well.

10. A doctor may close the case instance. The case instance reaches the state CLOSED.
3.3. Development methods to get a CMMN specification

3.3.1. Introduction

This chapter provides an answer to RQ3:

What are the development methods to get a CMMN specification?

Investigating existing literature did not deliver any specific methods for developing CMMN systems. There are no development methods to get a CMMN specification.

A few development methods exist for GSM, but considering the fundamental differences between GSM and CMMN (see 3.1.1) they cannot be used (completely) unless they are tuned for CMMN.

In the following two sections, two GSM-oriented development methods, which were a result of the literature study, will be shortly explained followed with a short conclusion.

In order to get a formulation of the requirements, based on the CMMN key concepts as defined in the CMMN meta model, one could make use of an input form document, derived from the standard, as defined in Appendix D ‘CMMN Form’. This form can be used in particular to specify the CMMN behaviour/plan model elements and inter-relationships. Although this form could be used to define the requirements by means of CMMN key concepts, it is strongly believed that from a practical perspective this is not possible without proper tooling which supports e.g. the composition and decomposition of the different key constructs and validation of required syntax.

In order to support business process reengineering a distinction can be made in business processes according to the as-is and to-be situation. Heuristics can be applied to optimise the artefact schemas and services as a result of steps 1 and 2. [16] defines several heuristics which can be applied for artefact-centric process models. This kind of business process optimisation is outside the scope of this study.

CMMN is artefact-centric. So the first idea was to look at the methods applied for other artefact-centric approaches to see if they can be partially used for CMMN modelling.

3.3.2. An artifact centric design methodology

In [1] an artifact-centric design methodology is proposed in which GSM-like schemas are constructed from scratch.

The design methodology follows a top-down approach and consists of the following steps:

STEP 1: Business Artifacts Discovery

(a) Identify critical artifacts for the business process
(b) Discover key stages of artifacts’ life cycles from the scenario-based requirements

STEP 2: Design of Business Operations Model (BOM)

(a) Logical design of artifact schemas
(b) Specify services for artifacts needed for moving artifacts through the life-cycles
(c) Develop ECA (Event Condition Action) rules that enable artifacts progress in their life cycles

STEP 3: Design of Conceptual Flow Diagram
STEP 4: Workflow Realization

The goal of step 1 is to develop a high level specification of the business operations through discovering key artifacts and important stages in their life cycles through a combination of top-down analysis and by examining typical scenarios.

Based on the basic skeleton produced in step 1, step 2 results in a preliminary design named the BOM (Business Operations Model). Step 2a focuses on data design of the artifacts. In step 2b abstract services are developed for the various business activities, which operate on the artifacts. In step 2c the associations between the services and the artifacts are specified.

Steps 3 and 4 start from the BOM with the goal of obtaining an operational workflow system. As a result of step 3 a conceptual flow diagram is delivered in which the associations between services and artifacts are procedural in nature. This result is further modified to satisfy the service behavioural constraints, and optimized according to performance metrics. These final steps are not in scope of this study as they are part of the run-time and not of the design.

It is possible to apply steps 1 and 2 partly for developing CMMN systems. A top-down approach is followed based on one or more scenario specifications and a further refinement of requirements and the design based on declarative constructs like stages, tasks, milestones and sentries can be partially achieved. Steps 3 and 4 are not feasible for CMMN systems because of the fundamental differences between GSM and CMMN as already explained in Chapter 3.

3.3.3. Synthesizing a GSM model from a Business Process Model

In [14] a semi-automated 2-phase transformation approach is defined for creating a GSM schema from a business process model.

Phase 1: creation of an object-centric design

The objective of phase 1 is to synthesize an object-centric design from a business process model that specifies the flow of multiple state full objects. The object centric design specifies in an imperative way the life cycle of the objects and the object interactions.

The first phase uses UML activity diagrams as input from which UML communicating state machines are translated specifying an object-centric design. The translation is based on synthesis rules. Each synthesis rule translates a basis substructure in an activity diagram to a basic substructure of a state machine. A distinction is made between synthesis rules that define:

- coordination logic for creation, finalisation, read-access and update-access of an object. Coordination logic rules result in an abstract object-centric design that is not executable, since it does not contain external event triggers or task invocations.

- execution logic. Each execution-logic synthesis rule translates a substructure, which contains a basic control-flow construct (task, decision, merge, fork, join) into an object-centric design.

Phase 2: creation of a data-centric system design

In phase 2 a mapping is done from an object-centric design to a declarative Guard-Stage-Milestone schema. This GSM schema can subsequently be refined into a complete specification of a data-centric system e.g. in CMMN notation.

Based on the following classification state machines are mapped to GSM schema:

- in a 'busy state' the system waits for an activity to complete (e.g. state 'busy completing order' of an object Order in an ordering process). Busy states are similar to stages.

- an 'object state' corresponds directly to an object node in the activity diagram (e.g. state 'paid' or 'unpaid' of an object Bill in an ordering process). Object states are similar to milestones.
Considering the fundamental differences between GSM and CMMN as already explained in Chapter 3, phase 2 cannot be applied for CMMN applications unless this part of the development method is tuned for CMMN. Besides this approach, which results in Phase 1 in a so-called object-centric design, can only be applied for CMMN applications if the current-state is modelled in UML activity diagrams.

The theoretical framework has provided some information for:
- interpretation and application of the CMMN standard and
- the steps to develop a CMMN design based on a top-down artifact-centric method: to define the requirements in terms of a scenario and use these scenario’s as input to define stages, milestones and other elements of CMMN.
4. Experiment at the sample patent office

4.1. Introduction

This chapter describes the experiment, which has been performed at the sample patent office in order to create some CMMN designs based on two practical cases.

In 4.2 ‘CMMN key concepts’ an answer is given to research question 4: which CMMN key concepts are relevant to test its practical application?

Thereafter it provides in 4.3. ‘Development approach’ an overview of the development approach which has been used to perform the experiment.

Subsequently in 4.4. ‘Experiment’, an answer is given to research question 5: what are the key requirements considered in the elicitation project which follows a model-driven development approach for a case management system at the sample patent office and how can they be used as test scenarios? This section describes the results of the transformation process, which has been used to derive a CMMN design from 2 scenario specifications.

Finally, 4.5 ‘Observations’ gives an overview of the major observations made during the performance of the experiment.

4.2. CMMN key concepts

This section gives answer to:

Research question 4: which CMMN key concepts are relevant to test its practical application?

The following set of CMMN development elements have been selected to test its practical application (see Chapter 3.2.3.2. ‘CMMN components semantics’):

- Case
- CaseFile
- CaseFileItem
- EventListener
- MileStone
- PlanFragment
- PlanItem and PlanItemDefinition
- Sentry
- Stage
- Task
- PlanningTable
- Behaviour property rules
- CaseRole,
- CaseParameter

In addition, the practical application of connectors, which is aesthetic, has been tested.
4.3. Development approach

For the experiment the data-centric design methodology of Bhattacharya [1] as specified in paragraph 3.3.2 has been partially applied (Step 1 and Step 2).

The development approach consisted of the following steps:

1. the patent application search process was analysed by means of a business process model (Petri net model),
2. three critical case file items were identified in this process namely ‘Patent application’, ‘Request for Clarification’ and ‘Clarification’ (Bhattacharya [1] Step 1a),
3. these case file items were relevant in two stages (‘sub processes’ in the process model) namely: ‘Division Allocation’ and ‘Search Validation’ (Bhattacharya [1] Step 1b),
4. for each stage the behavioural components were identified including sub-stages, milestones, tasks, events and sentries which were presented in a CMMN model (Bhattacharya [1] Step 2),
5. for CMMN modelling, an experimental software product was used from the company Trisotech named CMMN Web modeller [19]. Focus of the experiment was on design, not run-time execution as this was also not supported by the tooling.

4.4. Experiment

4.4.1. Introduction

In this section answer is given to:

Research question 5:

what are the key requirements found in the hypothetical elicitation project for the adoption of a case management system at the sample patent organisation and how can they be used as test scenarios?

The objective of the elicitation project is to define the business requirements for a new case management system, a necessity to support the sample patent office’s primary process limited to the search of patent applications.

In the following sections the results of the steps in the development approach are described.

In 4.4.2. ‘Experimental steps’ the results of the steps as part of the development approach are explained in context and further detail (in particular for step 1).

In 4.4.3. ‘Division allocation and Search validation’ the results of steps 3, 4 and 5 are worked out in more detail for stages ‘Division allocation’ and ‘Search validation’.

4.4.2. Experimental steps

4.4.2.1. Step 1: Initial business process as a workflow process

In order to define some practical cases, it was decided to follow the top-down development approach of Bhattacharya [1] after having first analysed the sample patent office’s search process, which starts after a patent application is filed (Development approach: step 1).

The objective was to define the sample patent office’s search process in a process model and thereafter identify, next to the patent application other business artifacts and sub-processes which could be used for testing the practical application of CMMN.
The results of this study, in particular the process model, are defined in Appendix E ‘Process model search process’. For the process modelling a Petri net like technique and notation has been used based on YAWL (Aalst & Hofstede, 2002) [17]. The notation of YAWL is described in Appendix F ‘YAWL notation’.

Figure 4-1 Part of the sample patent office’s search process, provides a part of the search process related to the allocation of a patent applicant to a division (group of examiners) and validation of the patent application against the rules. For further details, see Appendix E.

4.4.1.2. Step 2: Identification of critical business artifacts

In the second step of the development approach (Development approach: step 2) it was decided to follow Step 1a as defined in the top-down development approach of Bhattacharya [1] namely to identify critical business artifacts for the business process.

The following three critical business artifacts, were identified within the process model:
- a ‘Patent application’ being input for the search process,
- a ‘Request for clarification’ which can be sent by the patent examiner to the applicant when further information is required to handle the patent application before the search can take place and a search report will be created,
- ‘Clarification’ which is received by the sample patent office from the applicant as a result of a ‘Request for clarification’ send by the patent examiner.

**Identification of a case**

At the sample patent office the case ‘Dossier’ constitutes a CaseFile within the context of CMMN. This CaseFile, which is some kind of ‘folder’, is filled with the CaseFileItem (case file instance) ‘Patent application’ (is a request for a patent). Later it can be extended with other CaseFileItems like a ‘Request for clarification’ and a ‘Clarification’.

So a ‘Patent application’, ‘Request for clarification’ and ‘Clarification’ can be perceived as a CaseFileItemDefinition with their own instances (CaseFileItems).

With reference to CMMN, the important classes in the information model (see Information model in [6]) are:

- CaseFile: the container for all the information in the case instance,
- CaseFileItem: a piece of information in a case instance. All the CaseFileItems of a case instance are stored in a case’s CaseFile. A case instance may have a large number of CaseFileItems,
- CaseFileItemDefinition (not presented in Figure 3-2, see [6]): corresponds to the type of a CaseFileItem and defines the composition of the CaseFileItem by means of Properties,
- Property (note presented in Figure 3-2, see [6]): corresponds to a field of a CaseFileItem.

![Figure 4-2 Example of CMMN Information model elements](image-url)
Figure 4-2 Example of CMMN Information model elements represents an example CaseFile consisting of example CaseFileInstances of a ‘Patent application’, ‘Request for Clarification’ and ‘Clarification’.

4.4.1.3. Step 3: Discovery of key stages

Considering the three selected business artifacts it was decided to analyse in further details the sub-processes ‘Division allocation’ and ‘Search validation’ and to define these sub-processes as top level stages. For each stage a short scenario specification has been written to get a better understanding of the sub-process and to get at the right level of detail to identify ‘services’ (tasks) and ‘rules’ (sentries).

4.4.1.4. Step 4: Identification of sub stages, tasks, milestones, sentries etc.

Using the two scenario descriptions as input, sub-stages, tasks, milestones, sentries etc. were identified within the context of CMMN.

4.4.1.5. Step 5: Design

For each stage ‘Division allocation’ and ‘Search validation’ CMMN development elements were defined and structured using experimental software which delivered a design.

4.4.3. Division allocation and Search validation

4.4.3.1. Division Allocation

4.4.3.1.1. Scenario description Division Allocation

‘Division Allocation’ is the allocation of a patent application to a division. A division is a group of three persons within a directorate of the sample patent office, which play the role of 1st examiner, 2nd examiner and chairman. All three persons perform search process related activities, which are relevant for the patent application.

A more refined overview of ‘Division Allocation’ is presented in Figure 4-3 Process flow application from filing until allocation.

For the purpose of the experiment and the transformation to a CMMN model, in particular step 8 at the end is relevant. The other steps are presented to put ‘Division allocation’ in context with the steps before.
A patent application can be filed on paper or electronically. A paper application is received and scanned (step 1). The application can also be filed electronically (step 2). Subsequently the application is checked for formalities to guarantee information is complete and correct (step 4). In parallel the application is pre-classified which is required to allocate the application to the sample patent office’s search department responsible for the search in a certain technical field (step 5). The application is recorded in system 6 (step 6). In this system it is recorded if a pre-classification has been given (PRECLA) and/or formalities checking of the patent application against the rules of a convention has been taken place (FODO). The distribution of the application to the sample patent office’s department is handled in system 7 (step 7). Distribution can be based on knowledge of previous similar applications and/or based on an intellectually given pre-classification.

After a patent application has been distributed and allocated to the sample patent office’s department, Technical Acceptance (= acceptance that the patent application can be handled by the allocated search directorate) and Allocation to a division can take place (step 8).

The technical acceptance and allocation to a division is done intellectually by a case file manager. A case file manager is a dedicated patent examiner within a directorate which maintains the inventory of incoming patent applications. The case file manager allocates case files to divisions and if necessary re-distributes the incoming patent application to another directorate (or the same directorate) in case the patent application has not been pre-classified properly. This re-distribution can also be triggered by system 7 by setting a new PRECLA (not presented in Figure 4-3 Process flow application from filing until allocation).
The case file manager will subsequently allocate the patent application to a division. A Division consists of a 1st examiner, 2nd examiner and a chairman. The 1st examiner will take care for the search process.

4.4.3.1.2. Transformation Division Allocation

Division allocation is a stage containing a stage and a set of tasks related to the allocation of a patent application to a division (1st examiner, 2nd examiner and chairman) performed by a case file manager (Figure 4-4 Stage Division Allocation).

Figure 4-4 Stage Division Allocation

The first milestone of the stage ‘Division Allocation’ is ‘Technically Accepted’ given by a case file manager or a system which confirms that the Patent Application is accepted technically by the directorate for which the Patent Application has been pre-classified. Task and milestone are required (!). The task is also blocking.

The second milestone of the stage ‘Division Allocation’ is ‘Allocation Completed’. This milestone is achieved when all of the allocations (allocation of 1st examiner, 2nd examiner and chairman) of a division have been completed. Allocation tasks are executed by a case file manager. This stage is required (!) and repeatable (#) until complete allocation is achieved. It also has AutoComplete set to ‘true’ which means that the stage is perceived completed when the status of all tasks are set on completed. When ‘Allocation completed’ has been achieved persons allocated to a Patent Application are notified in task ‘Notify person allocated’.

The stage ‘Division Allocation’ can be terminated when the case file manager decides for redistribution of the patent application. Also for this stage an AutoComplete indicator is set to ‘true’, which closes the stage when no further planning or work is required any more. For redistribution the pre-classification of a patent application is changed and the patent application is returned to the distribution process. The human task ‘Redistribute’ is discretionary and can be defined by the case file manager in the planning at run-time.

4.4.3.2. Search Validation

4.4.3.2.1. Scenario description Search validation
Search validation is concerned with the validation of information items (for example claims) of the patent application with the articles and rules as formulated in a convention, see Figure 4-5 Search validation of a patent application for an example.

Before the search will start the application is validated for the way how the claims have been defined.

For an application for which a meaningful search can be carried out, the application is checked against Rule 62a which applies for patent applications containing a plurality of independent claims (see Figure 4-5 Search validation of a patent application).

The patent examiner discovers that the application does not meet Rule 62a and sends a Request for Clarification to the applicant or the examiner decides that the patent application meets Rule 62Aa. When ok the next rule is checked or the application is ready for a search.

After receipt of the Clarification the clarification is checked (see Figure 4-5) and the following two situations can start to exist:
- the applicant gives a response within the period of 2 months and the examiner decides to take over the response from the applicant. (see Figure 4-5 ‘ON TIME and OK?’). In this case the patent examiner will continue with validating the next rule or he will start the search.
- the applicant does not give a response within the period of 2 months (see Figure 4-5 ‘NOT ON TIME’) or the applicant gives a response within 2 months but the patent examiner decides not to consider the response from the applicant because it still does not meet the rule. (see Figure 4-5 ‘ON TIME and not OK?’). In this case the patent application procedure will be cancelled.

4.4.3.2.2. Transformation Search validation

Search Validation (Figure 4-6 Stage Search validation) is a stage containing a set of tasks to validate the compliance of the information provided by the applicant in the patent application against the rule(s) of the convention.
The stage ‘Search Validation’ is triggered by the allocation event of the 1st examiner. The stage is repeatable (#) for different rule checks.

The human task ‘Rules Compliance checks’ is required (!) and repeatable (#) and concerns the conformance of the patent application to the rules as defined in the convention.

Depending on the results, the patent examiner asks clarification or perceives the rule as validated and compliant which results in the achievement of milestone ‘Rule validated’. The stage ‘Clarification’ is optional. This stage is further decomposed (see Figure 4-7 Stage Clarification).

Stage ‘Clarification’ is a repeatable stage, which consists of two sub-stages:
- Stage ‘Prepare Request for Clarification’ and
- Stage ‘Respond to Clarification’.

Stage ‘Prepare Request for Clarification’ consists of two human tasks, which are required (!):
- Task ‘Create request for Clarification’ and
- Task ‘Send request for Clarification’ which sends a request for clarification to the applicant who has filed the patent application.
All stages have an AutoComplete indicator set on ‘true’.

Stage ‘Respond to Clarification’ can be triggered by an Event ‘Clarification received’ or the creation of a ‘Clarification’.

It consists of two tasks:
- a Human Task ‘Receipt of Clarification’ (Blocking) which must be completed before other tasks can progress,
- a Human Task ‘Review of Clarification’ (Blocking and Required) which makes an assessment weather the patent application is Compliant or Not-compliant to the rules which are milestones within the stage ‘Clarification’.

The response to the clarification is a Time-limited event and if that time is expired results in the termination of that stage.

When there is no-compliancy to a rule or when the time limit is expired a new task ‘Notify application of cancellation’ informs the applicant that his application will be cancelled.

4.5. Observations

During the experiment the following observations were made with respect to the design-time modelling:

1. Syntax rules.

The syntax rules for the CMMN design elements are not clearly described in the standard specification. Syntax rules and constraints are spread across the specification. This leads to less visibility and fragmentation of the rules. The syntax rules and constraints are not clear for the different levels of composition/decomposition of stages and planning models. There is no central rules management based on a detailed constraint language.

2. Behavior property rules.

The CMMN standard specification describes what the designer can do and what will be evaluated. Even the effects of evaluation are described. However, the aspect of how the rule should be constructed and how rich and complex this rule can be, is not explained. E.g. in stage ‘DIVISION ALLOCATION’ there is a case where the decision to have the task ‘REDISTRIBUTE’ in the planning depends on the content of a Property ‘PRE-CLASSIFICATION CODE’ as part of a caseFileItem ‘PATENT APPLICATION’ set by an external system (system 7) instead of an intellectual decision by a case file manager. In that case an applicability rule should be constructed to reflect this but it is not clear how this can be done. There is no specified order of precedence in the execution of the behavior property rules for the CMMN element (e.g. tasks or stages) for which the rules applies, and there is no way to set this order at design time.

3. Multi-case control.

The design of the CMMN meta model is limited to the definition of caseFileItemDefinitions which relate to one Case. In CMMN conditions (e.g. for stages and milestones) can refer only to attributes within a single Case instance. This could impose a limitation in the expression of the design and its implementation.

4. Notation.

During the design of the practical application as part of the experiment, the representation of tasks to their respective roles using a swim lane type design approach has been omitted in CMMN as there is no visual representation of roles. Although the practical application as part of the experiment is limited to only a few roles the complete search process at the sample
patent office is considerably more complex. In this case the declarative approach to the allocation of tasks to roles would be much clearer and more transparent during the design.

The model is missing the ability to capture annotations to the elements of the CMMN model as well as being able to make comments or notes in relation to the model. Just the use of so-called ‘Connectors’ is too limited to get understanding of the model.

It is unclear how milestones should be visualized which are part of the case plan model and which relate and appear outside a sub stage.

5. Roles.

It is not clear what type of roles (e.g. account role, assignment role etc.) are applicable within the context of CMMN and the effect on the model in terms of tasks, permissions and authorisations. For example, the task ‘ALLOCATE 2nd EXAMINER’ in stage ‘ALLOCATION’ could be executed by a case file manager but also another role under different conditions. It is not clear how this should be designed and there is no expression language to accommodate these type of conditions.

6. Practical application.

The experiment has delivered a design which covers in particular the information model and behavior model. It proves that the CMMN model offers a defined framework for specifying the case management requirements. During the experiment it was experienced that the design can be extended in a flexible way by adding new caseFileIdemDefinitions and other development elements. This was even possible after having untouched the design for a long period. It was also clear where changes were made to the practical application what parts of the model were impacted in the design.

7. Terminology.

CMMN element terminology is sometimes not clear and ambiguous. In some cases it is also inconsistent. E.g. it is not clear in the standard specification what is meant with a Task type = ‘none’ and why there is a RequiredRule for an eventlistener.


Although there are fundamental differences between GSM and CMMN, it was possible to partially follow the top-down artifact-centric design methodology of Bhattacharya [1].
## 5. Conclusions and recommendations

Table 5-1 Conclusions and recommendations lists the conclusions and recommendations as a result of this research.

<table>
<thead>
<tr>
<th>Conclusions</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax rules are not completely clear</td>
<td>Enrich the constraint language</td>
</tr>
<tr>
<td>It is not clear how behavior property rules should be implemented. The precedence of execution is not specified.</td>
<td>Definition of rules and development of guidelines how to implement behavior property rules</td>
</tr>
<tr>
<td>No options to support the design of applications with multiple cases.</td>
<td>Extension of the CMMN meta model.</td>
</tr>
<tr>
<td>No swim-lane type notations are available to easily allocate tasks to roles</td>
<td>The visual representation of roles should be improved in the standard (e.g. by means of swim-lanes)</td>
</tr>
<tr>
<td>The expressive features in the notation are not enough to have a complete understanding of the CMMN Model.</td>
<td>The meta model should be extended to include the possibility of annotations, comments and notes.</td>
</tr>
<tr>
<td>The type of roles (e.g. account role, assignment role etc.) and the effect on the model in terms of tasks and permissions is not clear. Where there are multiple roles that can be applied, there is no way of specifying the conditions for when they are applicable.</td>
<td>The meta model should extend the type of roles and related conditions as to when those roles apply using expressions.</td>
</tr>
<tr>
<td>The experiment showed that it is possible to create a CMMN design and to show the benefits of CMMN. Applying the CMMN standard will strengthen the uniform specification of case management requirements by means of Lego® style building blocks</td>
<td>Compliance to the CMMN Information and behavior model and related lifecycle concepts will promote the recognition of the standard.</td>
</tr>
<tr>
<td>Although it could not be concluded from the experiment, the declarative aspects of CMMN creates a shorter timeline between design and execution delivering significant benefits in speed and understanding</td>
<td>An agile development approach to make use of the prototyping and simulation strengths of CMMN</td>
</tr>
<tr>
<td>CMMN element terminology is sometimes not clear and ambiguous. In some cases it is also inconsistent.</td>
<td>Improvement of terminology.</td>
</tr>
<tr>
<td>There is a lack of specific CMMN methodologies. The experiment proves that a GSM data-centric methodology can be used to construct a CMMN reference model</td>
<td>Due to the conceptual differences between GSM and CMMN the methodology used should be tuned to CMMN (i.e. planning model) and promoted as a best practice</td>
</tr>
</tbody>
</table>

**Table 5-1 Conclusions and recommendations**

Considering above the central research question:

**Can CMMN be applied in practise?**

can be answered as follows:

**Yes.**
The experiment has proved that it is possible to deliver a CMMN design which covers the CMMN elements as defined in the CMMN meta model. It also proves that the CMMN standard specification [6] offers a defined framework for specifying case management requirements.

However, considering the observations, conclusions and recommendations the standard is still not mature enough and should be improved based on additional research as recommended.

The completeness of the standard in its current form is sufficient, in order to confirm that the CMMN standard can be used. The standard at the moment allows room in certain areas (as indicated in the observations) for a variety of interpretations and this should be limited to avoid divergent implementations. Therefore reference applications are required to put the CMMN standard into practice.
6. Research Process Reflection

For research question 5, the hypothetical elicitation project deliverables are required to be mature and complete enough to serve as a realistic driver for the experiment. In considering the realisation of case management at the hypothetical patent office, it was deemed essential that it should follow a structured approach to implement case management.

Moreover it needs to be mentioned that the sample patent office’s development approach is required to ensure that the key concepts and semantics of the case management development model are clearly defined and should follow a life cycle concept.

Within the scope of the theoretical framework no CMMN specific development approaches were available. Instead development approaches focussed on GSM models were considered as specified within the theoretical framework. Although there were only two selected, it is possible there may be other similar approaches existing that were not considered.

The research was restricted to the design-time modelling of a CMMN application. Next to the absence of appropriate tooling, this was the reason why the run-time execution of a CMMN implementation has not been tested.

The reliability of the research is based on the evidence of the participants involved in the experiment. Especially given the fact that no specific CMMN development approaches are available and the standard is the only authority as to how the CMMN elements should be applied in practice.

In respect to the quality of the CMMN models the following can be stated:

- for the terminology it is important to use a maintained glossary of terms,
- to assure the quality of the existing process, the development of the Petri-net business process model took place together with a patent examiner,
- to assure the quality of the scenario descriptions a business analyst and solution architect were involved,
- the CMMN model scope and precision were sufficient and complete enough to cover all the relevant CMMN elements,
- as far as the standard allowed, the internal consistency of the CMMN model was checked. Observations and recommendations have been made on the formulation and enrichment of the CMMN syntax rules,
- the integration of the CMMN models with models in other notation(s) was not covered,
- the external consistency of the CMMN models was not judged or assessed.
References


Appendix A. Sample patent office

Organisation

The sample patent office is a hypothetical patent office providing a uniform application procedure to enable the filing and registration of patents.

For the purpose of the practical application the sample patent office is perceived to be distributed over different locations.

Stakeholders

The sample patent office’s external services are provided across four broad categories of external customers (or stakeholders):

1. Patent departments,
2. IP Professionals (inventors, applicants, patent owners, patent attorneys),
3. Public and organisational bodies (members of public, academics, economists, non government organisations):
4. Policy, standards bodies and groups.

These groups of external customers have one or more of the following reasons for interacting with the sample patent organisation and the patent system:

- To use patent information to inform and drive innovation;
- To protect innovation using a patent application;
- To realise value from a patent / patent application and
- To reduce costs / drive economic growth.

Primary process

Depending on the type of application different business processes can be identified. The overall primary business process to grant patents and adjudicate appeals can be specified in the following main processes and sub processes:

- Grant patent rights:
  - Receive application,
  - Search,
  - Examination,
  - Opposition,
- Publish register,
- Classify prior-art,
- Manage payments,
- Other operational support,
- Adjudicate appeals.

Patent granting, opposition and appeal procedures are fully standardised at the sample patent office and consist of different steps. The sample patent office charges fees for handling these steps. The patent applicant can decide at any time to abandon the procedure by not paying a fee, which is due; the sample patent office then deems the application to be withdrawn. In most cases, the sample patent office does not start performing the service related to each step until the applicant has paid the fees involved.
Appendix B. GSM (Guard Stage Milestone) semantics

Figure B-1 Semantic GSM Artifact-centric model presents a GSM meta model, presented as a Entity Relationship Model, which has been derived from a semantic GSM model defined in [20]. The boxes represent entities and the diamonds represent the relationships between the entities. The diagram also presents cardinalities between and specialisations of entities (arrows).

Business objects represent the data attributes as defined in the information model, while status objects (and relationships between them) represent the lifecycle. We see that the standard operates with such notions as an event instance, that can be sent and received by the artifact and that there are tasks triggered by the event which can change the status of the artifact.

![Semantic GSM Artifact-centric model](image)

**Figure B-1 Semantic GSM Artifact-centric model**

Figure B-2 Example GSM implementation presents an example GSM implementation of Figure B-1 Semantic GSM Artifact-centric model.
The following sections will provide an informal and intuitive introduction to the GSM theoretical conceptual meta model and its semantics as presented in Figure B-1 Semantic GSM Artifact-centric model and implemented as an example in Figure B-2 Example GSM implementation.

The GSM meta model is based on a set of declarative rules that dynamically generate an explicit sequencing of the various atomic tasks within the process/lifecycle, while naturally allowing for a hierarchical definition of the processes [2]. GSM is based on the Business Entity with Lifecycle (BEL) model [11].

GSM can be viewed, as a procedural system that permits the use of a rich rule-based paradigm for determining, at any moment in time, what activities should be performed next. It is intended to support the management of business-related activities, but not support the details of executing those activities.

The objective of the GSM meta model is on the one hand to have the business level constructs incorporated in an intuitive and flexible manner, and on the other hand to support a precise semantics that can support both implementation and mathematical investigation.

The objectives and expected benefits to specify business operations and processes based on the GSM meta model are:
1. To help business-level stakeholders to gain insight and understanding into their business operations,

2. To have a model that is centred around intuitively natural constructs that correspond closely to how business-level stakeholders think about their business,

3. To provide a high-level, abstract view of the operations, and gracefully incorporate enough detail to be executable,

4. To support a spectrum of styles for specifying business operations and processes, from the highly "prescriptive" (as found in e.g. BPMN) to the highly "descriptive" (as found in Adaptive Case Management systems),

5. To serve as a target into which intuitive, informal, and imprecise specifications and constraints of the business operations (e.g. in terms of "business scenarios") can be mapped.

The semantics for GSM B-steps has different formulations each with their own value [12] but in the context of this study the incremental formulation of the operational semantics will be used which corresponds roughly to the incremental application of the ECA like rules, providing an intuitive way to describe the operational semantics of a GSM model, and providing a natural, direct approach for implementing GSM.

There are several constructs in the GSM meta model of which four constructs are really four key constructs namely Information model, stage, milestone and guard.

Each of the GSM constructs, including the four key constructs, is described below.

(a) Artifact instance:

An artifact instance (see Figure B-1 Semantic GSM Artifact-centric model) corresponds intuitively to a single business-relevant conceptual entity of a particular artifact type (or business object) in a given domain, which combines both static properties, describing the data of interest, and the dynamics, induced by processes that manipulate such data as it progresses through some business operations.

An example of an artifact instance is a Patent application, which is created by an Applicant, which is subsequently checked by a Patent Formality Officer, and Patent Examiner at a Patent Office for formalities and payment and which is finally recorded in the search collection after a patent for the Patent application has been granted.

(b) Environment:

This corresponds to the environment within which artifact instances exist. The environment hosts the task executions that are invoked by artifact instances.

(c) Information model:

The Information model (see Figure B-1 and Figure B-2) provides an integrated view of all business-relevant information about an artifact instance.

The Information model contains 3 types of attributes:

- Data attributes:

  Data attributes are business-relevant data about a given artefact type (or related business object) as it moves through the business. Data attributes hold data about the business itself and how it is being affected by the artifact instance.

  There is at least one data attribute, which holds the unique, immutable identifier (id) of the artifact instance.
In Figure B-2 ‘da1’ until ‘da6’ are examples of data attributes. ‘da3a’ and ‘da3b’ present examples of data attributes of the same data type.

- **2 Event occurrence bookkeeping attributes:**

  These attributes are:
  - Most recent Event type and
  - Most recent Event time related to the incoming event occurrence, which is currently affecting or has most recently affected the business artifact instance.

  In Figure B-2 these attributes are presented as ‘mre type’ and ‘mre time’.

- **Status attributes:**

  Status attributes are holding ‘control information’ about:
  
  a) the current status of all milestones (‘true’ or ‘false’) and all stages (‘open’ or ‘closed’),
  
  b) for each milestone or stage the timestamp of the most recent change of status.

  Milestones and stages are described below in (d) and (e).

  In Figure B-2 ‘md1’ until ‘md6’ are examples of status attributes related to milestones ‘m1’ until ‘m6’ presented as circles in the same model and ‘sd1’ until ‘sd3’ examples of status attributes related to stages ‘s1’ until ‘s3’ presented as rounded boxes in the same model.

(d) **Milestone:**

A milestone is a business-relevant objective (at different levels of granularity) that can be “achieved” and may be “invalidated” by an artifact instance. It is a status object and part of the Life cycle model of an artifact instance.

Milestones are controlled in a declarative manner, based on triggering events and/or conditions.

Milestones are presented in Figure B-1 and in Figure B-2 as circles.

(e) **Stage:**

A stage is a cluster of business-relevant activity that might be performed for, with, and/or by an artifact instance, in order to achieve one (or more) of the milestones owned by that stage. It is a status object and part of the Life cycle model of an artefact instance.

Each milestone corresponds to one alternative way that the stage might reach completion. A stage becomes “closed” (or “inactive”) when one of its milestones is achieved. At most one milestone of a stage can be true at a time.

Stages may be nested (hierarchy) in ‘sub stages’ (see Figure B-1, role ‘substage’) creating a forest, to provide a rich form of modularity for specifying the overall behaviour of an artifact instance. The roots of this forest are called ‘top-level stages’, and the leaves are called ‘atomic Stages’. Stages are presented in Figure B-2 as rounded boxes. Stages ‘s2’ and ‘s3’ are sub stages of stage ‘s1’.

Each atomic stage contains exactly one task.
Stages can be *child* (descendant) and/or *parent* (ancestor) from each other. A non-leave node is called a *composite stage*.

A stage may execute multiple times in sequence, but can't have two occurrences that are executing simultaneously.

(f) **Guard:**

A guard is used to control whether a stage becomes “open” (or “active”) or “closed” (see Figure B-1).

A stage has one guard and is, like milestones, controlled in a declarative manner, based on triggering events and/or conditions. Each guard is specified as a sentry. Guards are unnamed, and their status cannot be referred to.

Guards are presented in Figure B-2 as diamonds.

(g) **Task:**

A task is a function from the atomic stages. A task corresponds to a unit of business-relevant work that is to be performed by an outside agent (either human or machine).

Tasks are invoked by artifact instances. When a task is invoked, the artifact instance provides input data from its Information model, and when the task terminates the task output is written into the artifact instance Information model.

Artifact systems interact with the environment based on two categories of tasks:
1. Invoking a 2-way service call,
2. Sending a one-way message.

(h) **Event:**

In GSM there are three categories of event:

1. **Incoming event:**

These are events that can be send from the environment to artifact instances. There are 2 kinds of incoming event:

1) **One-way message event** (also referred to as *request event*), which corresponds to a message sent pro-actively from the environment, e.g. a user-request,

2) **Task termination event**, which corresponds to the message sent into an artifact instance upon termination of a previously invoked task,

2. **Internal event** (also referred to as *status change event*).

These correspond to when a milestone or stage changes status.

3. **Outgoing event:**

These are events sent from artifact instances to the environment. There is one kind of outgoing event, namely *task invocation event*, which correspond to when an artifact instance invokes an occurrence of a task.
A sentry is an expression in a condition language that can refer to incoming events, internal events, and the values of data and status attributes as defined in the Information model.

Incoming Events can be:
- Return calls from 2-way service calls,
- One-way messages,
- Requests that a new artifact instance is to be created.

Sentries govern the progress of an artifact instance. In practical settings, a sentry will have the form: on <event> if <condition> then <action>, where either the event or condition may be omitted.

In other words, a sentry is achieved and becomes true if an appropriate triggering event (incoming or internal) occurs and/or the condition becomes true.

A sentry is used:
- As guard, to control when a stage opens (see Figure B-1, role ‘guards’), and
- To control when a milestone is achieved (achieving sentry) or invalidated/become false (invalidating sentry) (see Figure B-1, role ‘achv’).

The Life cycle model (see Figure B-1 and Figure B-2) is a component that specifies the milestones and stages of a GSM artifact model, including their relationships (stage hierarchy, association of milestones to stages, and association of tasks to atomic stages), and the sentries that govern the guards and milestones.

In Figure B-1, an artifact type is a particular business object in that it has a lifecycle, i.e., a set of status objects connected to itself through the SORefersTo role. A status object is either a milestone or a stage and this specialisation is disjoint and complete.

The key concepts for the GSM operational semantics are:

(a) **Processing of incoming events**: Incoming events are processed one at a time. E.g. with reference to Figure B-2, for a single artifact and a single stage s1, that stage s1 can have at most one occurrence that is executing at one time.

(b) **GSM invariants**: stages and milestones satisfy two properties that correspond to central business-level intuitions concerning how they interact. These are:
   - GSM-1: If a stage S owns a milestone m, then it cannot happen that both S is active and m has status true. If S becomes active then m must change status to false, and if m changes status to true then S must become inactive.
   - GSM-2: If stage S becomes inactive, the executions of all sub stages of S also become inactive.

(c) **Snapshot**: a snapshot is an instantaneous description of an artifact instance at some moment in time. An artifact instance can be perceived as a sequence of snapshots that it moves through during its lifetime.

(d) **Business step (B-step)**: a B-step corresponds to the atomic unit of business-relevant processing in a GSM model. It corresponds to the smallest unit of business-relevant change that can occur to a GSM system. It relates to what happens to a snapshot (all relevant aspects of a GSM system at a given moment of time) when a single incoming event e.g. from a user, from another system, or from the completion of a previously invoked task, is incorporated into the system. The focus is on what happens sequentially to the key elements like Information model, milestones, stages and guard as a result of an incoming event. What stages are opened and closed, and what milestones are achieved (or invalidated) as a result of an incoming event.
Appendix C. CMMN B-step

Figure C-1 Interaction scheme for B-step provides a graphical presentation of an example B-step within a stage derived from [13] page 6 Example 2.1. It presents the possible interactions between the different CMMN (GSM) components and to understand the figure it is important to read [13].

In order to show the many-many ‘process-flows’, Figure C-2 Sample CMMN (GSM) interaction scheme provides a graphical presentation of all possible interactions between the different CMMN (GSM) components as defined within a stage. It is clear that an event can cause a ‘complex’ escalation of interactions between the different components, which makes it not easy to test CMMN applications.
Figure C-1 Interaction scheme for B-step
Figure C-2 Sample CMMN (GSM) interaction scheme
Appendix D. CMMN form

This appendix describes a form (template), which can be used to prepare a specification according to the CMMN (Case Management Model and Notation) standard version 1.0 as published by the OMG (Object Management Group) [6]. The standard is available on http://www.omg.org/spec/CMMN/1.0/.

The document consists of a template specification and diagrams, which is structured according to the grouping of CMMN design elements as defined in chapter 3 of this document and as described in more detail in Chapter 5 of the CMMN notation standard as mentioned above.

The scope of the form is limited to the elements which fit the behaviour model/plan model of CMMN.

At the end of the appendix a table provides an overview of definitions of the Case management model elements presented in the diagrams. Other terminology and a graphical overview of the different CMMN elements can be found in this thesis and the CMMN standard itself.
1. CMMN Base declarations
2. CMMN Case elements

case

    name       String

role

    name       String
3. Plan Item Control

planItemControl

Abstract class, specifying defaults for aspects of control of PlanItemDefinitions

Relationships:
(a) repetitionRule: repetitionRule [0..1]
(b) requiredRule: requiredRule [0..1]
(c) manualActivationRule: ManualActivationRule [0..1]

A planItemControl MUST contain at least one RepetitionRule (a) or one RequiredRule (b) or one ManualActivationRule (c).

(d) A PlanItemControl MUST be the itemControl of:
- a PlanItem
- a DiscretionaryItem
- defaultControl of a PlanItemDefinition.

repetitionRule

name String

Relationships:
(a) condition: Expression [1] that MUST evaluate to Boolean.
True: instance of Task or Stage or MileStone maybe repeated.  
False: instance of Task or Stage or MileStone may not be repeated.

(b) contextRef: CaseFileItem[0..1]

CaseFileItem that serves as starting point for evaluation of the Expression that is specified by the condition of the RepetitionRule. If not specified, evaluation starts at the CaseFile object that is referenced by the Case as its caseFileModel.

requiredRule

name: String

Relationships: 
(a) condition: Expression [1]  
that MUST evaluate to Boolean.

True: instance of the Task or Stage or MileStone is required and MUST be in state Disabled, Completed, Terminated or Failed before its containing Stage (instance) can complete.
False: instance of the Task or Stage or MileStone is considered optional before its containing Stage (instance) can complete.

(b) contextRef: CaseFileItem[0..1]

CaseFileItem that serves as starting point for evaluation of the Expression that is specified by the condition of the RequiredRule. If not specified, evaluation starts at the CaseFile object that is referenced by the Case as its caseFileModel.

manualActivationRule

name: String

Relationships: 
(a) condition: Expression [1]  
that MUST evaluate to Boolean.

True: instance of Task or Stage is activated automatically when it is in state Available  
False: instance of Task or Stage is activated manually when in state Enabled

(b) contextRef: CaseFileItem[0..1]

CaseFileItem that serves as starting point for evaluation of the Expression that is specified by the condition of the ManualActivationRule. If not specified, evaluation starts at the CaseFile object that is referenced by the Case as its caseFileModel.
4. CMMN Information Model (or Case File)

caseFile

caseFileItem

caseFileItemDefinition

property

expression

language

URI

language in which the expression is specified.

body

string
5. Plan Item Definition

planItemDefinition

Abstract, specialised in several concepts:
- Planfragment (Stage),
- Task,
- EventListener,
- Milestone.

name String

Relationships:
(a) defaultControl: PlanItemControl[0..1].
DefaultControl MUST NOT be specified for the Stage that is referenced by the Case as its casePlanModel.

planFragment

(name)
String

See 2.9 "Plan Fragment and Stage".

task

(name)
String

(manualRule) Boolean (yes or no)
(repetitionRule) Boolean (yes or no)
(requiredRule) Boolean (yes or no)
(discretionary) Boolean (yes or no)

See 2.6 "Tasks".
**eventListener**

Abstract class.

*(name)* String

See 2.7 "Event elements".

**mileStone**

*(name)* String
6. Task elements

**task**

(task type) One of the following options: humanTask, processTask, caseTask, None. See below.

isBlocking Boolean (yes or no)

True: Task is waiting until the work associated with the Task is completed. MUST be default value.
False: Task completes immediately, upon instantiation. MUST NOT have outputs

Relationships: (a) inputs: CaseParameters[0..*]

(b) outputs: CaseParameters[0..*]

**humanTask**

Relationships: (a) planningTable: PlanningTable[0..1]

Optional. The HumanTask can be used for planning, and its PlanningTable might contain TableItems that are useful in the particular planning context. See page 36 and 37 of the CMMN standard.
**processTask**

Relationships:
- (a) processRef: Process[1]
  Reference to a Process
- (b) mappings: ParameterMapping[0..*]

**caseTask**

Relationships:
- (a) caseRef: Case[1]
  Reference to the Case that is called as part of the CaseTask.
- (b) mappings: ParameterMapping[0..*]

**process**

- name: String
- implementationType: URI
  The implementation type of the Business process. It MUST be provided in URI format. BPMN2.0, XPDL2, WSBPEL2 or WSBPEL1.

Relationships:
- (a) inputs: ProcessParameter[0..*]
  Zero or more inputs of the Business process
- (b) outputs: ProcessParameter[0..*]
  Zero or more outputs of the Business process

**parameter**

Abstract class, specialised in 2 concepts:
- processParameter,
- caseParameter.

**processParameter**

(name): String

**caseParameter**

(name): String

Relationships:
- (a) bindingRef: CaseFileItem[0..1]
  A reference to a CaseFileItem. The effect of the execution of instances of this Task on instances of the referenced CaseFileItem can be observed in terms of transitions in the CMMN-lifecycle of that CaseFileItem.
- (b) bindingRefinement: Expression[0..1]
An optional Expression to further refine the binding of the CaseParameter to the CaseFileItem, that is referenced. See page 36 CMMN standard.

**parameterMapping**

Relationships:

(a) sourceRef: Parameter[1]

(b) targetRef: Parameter[1]

(c) transformation: Expression[0..1]

Transformation language
7. Event elements

eventListener

- specialisation in 2 other concepts:
  - UserEventListener,
  - TimeEventListener.

userEventListener

Relationships:
- (a) authorisedRoleRefs: Role[0..*]
  Roles authorised to raise user event.

timerEventListener

- timeExpression: string
  expression string conforming ISO-8601 format for date and time, duration or interval representation.

Relationships:
- (a) timerStart: StartTrigger[0..1]
  Optional. The starting trigger of the TimerEventListener. If timerStart is specified, then at runtime, if the trigger occurs the time of occurrence of the trigger is captured and the timerExpression SHOULD be relative to the timestamp captured when the timerStart trigger occurs.

startTrigger

Abstract class that implements 2 concepts:
- CaseFileItemStartTrigger,
  -PlanItemStartTrigger.

**caseFileItemsStartTrigger**

standardEvent CaseFileItemTransition

Relationship: (a) sourceRef: CaseFileItem[1]

**planItemsStartTrigger**

standardEvent PlanItemTransition

Relationship: (a) sourceRef: PlanItem[0..1]
8. Sentry

sentry

  name: String

  Relationships:
    (a) onParts: OnPart[0..*]
    (b) ifPart: IfPart[0..1]

onPart

  Abstract class, specialised in 2 concepts:
  - CaseFileItemOnPart
  - PlanItemOnPart.
  See below.

caseFileItemOnPart

  standardEvent: CaseFileItemTransition

  Reference to a state transition in the CaseFileItem lifecycle.
  See table CaseFileItemTransition.

  Relationships:
    (a) sourceRef: CaseFileItem[1].
    Reference to a CaseFileItem undergoing the state transition as specified, the OnPart MUST occur (in run-time).

planItemOnPart

  standardEvent: PlanItemTransition

  Reference to a state transition in the PlanItem (casePlanModel, Stage, Task, EventListener or MileStone).
  See table PlanItemTransition. As different states apply to different PlanItems check also Table 5.23 in CMMN standard.

  Relationships:
    (a) sourceRef: PlanItem[0..1].
Reference to a PlanItem undergoing the state transition as specified, the OnPart MUST occur (in run-time).

SourceRef represents a PlanItem that MUST be contained by the same PlanFragment (or Stage) that also contains the Sentry that contains the PlanItemOnPart.

(b) sentryRef: Sentry[0..1].

Reference to a Sentry. It enforces that the PlanItemOnPart of the Sentry occurs when the PlanItem that is referenced by sourceRef transits the exit transition in its lifecycle, due to the Sentry that is referenced by sentryRef being satisfied.

SentryRef, if specified, MUST refer to a Sentry that is referenced by an exitCriteriaRef of the PlanItem that is referred to as the sourceRef of the PlanItemOnPart.

When sentryRef is specified, standardEvent MUST have value “exit”.

ifPart

(a) contextRef: CaseFileItem[0..1].
CaseFileItem that serves as starting point for evaluation of the Expression. If not specified, evaluation starts at the CaseFile object that is referenced by the Case as its caseFileModel.

(b) condition: Expression[1]
must evaluate to Boolean.
9. Plan Fragment and Stage

**planFragment**

Root element for PlanItems that should go into the plan as “a unit”.

Relationships:

(a) planItems: PlanItem [0..*]

(b) sentries: Sentry [0..*]

**planItem**

name

String

Relationships:

(a) itemControl: PlanItemControl[0..1]

If a PlanItemControl object is specified for a PlanItem, then it MUST overwrite the PlanItemControl object of the associated PlanItemDefinition element. Otherwise, the behavior of the PlanItem object is specified by the PlanItemControl object of its associated PlanItemDefinition.

(b) definitionRef: PlanItemDefinition[1]

DefinitionRef MUST NOT represent:
- the Stage that is the casePlanModel of the Case,
- a PlanFragment that is not a Stage.

DefinitionRef of a PlanItem that is contained by a Stage MUST NOT be that Stage or any Stage in which that Stage is nested.

(c) entryCriteriaRefs: Sentry[0..*]

MUST refer to Sentries that are contained by the Stage or PlanFragment that contains that PlanItem.

A PlanItem that is defined by an EventListener MUST NOT have entryCriteriaRefs.
(d) exitCriteriaRefs: Sentry[0..*]

MUST refer to Sentries that are contained by the Stage or PlanFragment that contains that PlanItem.

A PlanItem that is defined by an EventListener or defined by a Task that is non-blocking (isBlocking set to “false”) MUST NOT have exitCriteriaRefs.

Stage instance completion criteria are as follows:

1. **autoComplete = true**
   There are no Active children, AND all required (requiredRule evaluates to “true”) children are in {Disabled, Completed, Terminated, Failed}.

2. **autoComplete = false**
   There are no Active children AND (all children are in {Disabled, Completed, Terminated, Failed} AND there are no DiscretionaryItems) OR (Manual Completion AND all required (requiredRule evaluates to “true”) children are in {Disabled, Completed, Terminated, Failed}).
In other words, a Stage instance SHOULD complete if a user has no option to do further planning or work with the Stage instance.

If “false”, a Stage requires a user to manually complete it, which is often appropriate for Stages that contain “discretionary” items and/or non-required Tasks or Stages.

Question: is this logically correct???

Relationships:

(a) exitCriteriaRefs: Sentry[0..*]

Only the Stage that is referenced by the Case as its casePlanningModel can have exitCriteriaRefs. Note that it is only useful for that Stage to directly have exitCriteriaRefs, as it cannot be further nested in other Stages (other Stages can contain both PlanItems that represent Stages and the Sentries that impose entry and/or exit criteria on them).

Question: no entryCriteriaRefs???

(b) planningTable [0,1]  Optional

(c) planItemDefinition [0..*]

PlanItemDefinitions MUST NOT be contained by any other Stage then the casePlanningModel of the Case.
10. Planning Table

**tableItem**

Relationships:

(a) `authorisedRoleRefs: Role[0..*]`

References to zero or more Role objects that are authorised to plan based on the TableItem.

(b) `applicabilityRules: ApplicabilityRule[0..*]`

If the condition of the ApplicabilityRule object evaluates to “true”, then the TableItem is applicable for planning, otherwise it is not. If no ApplicabilityRule is associated with a TableItem, its applicability is considered “true”.

A PlanningTable that contains a TableItem MUST contain the ApplicabilityRules that represent the `applicabilityRuleRefs` of that TableItem.

**planningTable**

Relationships:

(a) `tableItems: TableItem[0..*]`

PlanningTable can be “nested”.

(b) `applicabilityRules: ApplicabilityRule[0..*]`

**discretionaryItem**

Relationships:

(a) `planItemDefinitions: PlanItemDefinition[1]`

Represents a Task or a PlanFragment (or Stage).

(b) `itemControl: PlanItemControl[0..1]`

optional. Overwrites the value of attribute `defaultControl` of the DiscretionaryItem associated PlanItemDefinition.
(c) entryCriteriaRefs: Sentry[0..*]
Reference to zero or more Sentries that represent the entry criteria

(d) exitCriteriaRefs: Sentry[0..*]
Reference to zero or more Sentries that represent the exit criteria.

A DiscretionaryItem that is defined by a Task that is non-blocking (isBlocking set to “false”) MUST NOT have exitCriteriaRefs.

**applicabilityRule**

name  String

Relationships:

(a) condition: Expression[1]
that MUST evaluate to Boolean.

True: TableItem is available for planning.

(b) contextRef: CaseFileItem[0..1]
CaseFileItem that serves as starting point for evaluation of the Expression that is specified by the condition of the ApplicabilityRule. If not specified, evaluation starts at the CaseFile object that is referenced by the Case as its caseFileModel.
### Overview and definition of Case Management Model elements in alphabetical order

<table>
<thead>
<tr>
<th>Case Management Model element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>applicabilityRule</td>
<td>Are used to specify whether a TableItem is &quot;applicable&quot; (&quot;eligible&quot;, &quot;available&quot;) for planning, based on conditions that are evaluated over information in the CaseFile.</td>
</tr>
<tr>
<td>caseFileItemOnPart</td>
<td>Inherits from OnPart.</td>
</tr>
<tr>
<td>caseParameter</td>
<td>Used to model the inputs and outputs of Cases and Tasks</td>
</tr>
<tr>
<td>caseTask</td>
<td>Can be used to call another Case. A CaseTask triggers the creation of an instance of that other Case, which creation denotes the initial transition in the CMMN-defined lifecycle of a Case instance.</td>
</tr>
<tr>
<td>discretionaryItem</td>
<td>Identifies a PlanItemDefinition, of which instances can be planned, to the &quot;discretion&quot; of a Case worker that is involved in planning, which instances are planned into the context that is implied by the PlanningTable that contains the DiscretionaryItem, either directly, or via a 'nested' PlanningTable. See standard Fig. 4.1 – Design time phase modelling and run-time phase planning. The conditions for a PlanItemDefinition or Stage to be &quot;discretionary&quot; are explained in the standard.</td>
</tr>
<tr>
<td>eventListener</td>
<td>An event is something that &quot;happens&quot; during the course of a Case. Distinction is made between &quot;standard events&quot; that denote transitions in the CMMN-defined lifecycle of CaseFileItems and transitions in the CMMN-defined lifecycle of Stages, Tasks and Milestones.</td>
</tr>
<tr>
<td>expression</td>
<td>Specifies string objects that are evaluated over information in the CaseFile.</td>
</tr>
<tr>
<td>humanTask</td>
<td>Unit of work performed by a Case worker.</td>
</tr>
<tr>
<td>ifPart</td>
<td>Used to specify an (optional) condition.</td>
</tr>
<tr>
<td>manualActivationRule</td>
<td>Specifies under which conditions Tasks and Stages, once enabled, start manually or automatically. It determines whether the Task or Stage instance should move to state Enabled or Active. This rule is evaluated when one of the entry criteria of the Task or Stage is satisfied. If this rule evaluates to &quot;true&quot;, the Task or Stage instance transitions from Available to Enabled, otherwise it transitions from Available to Active. This rule impacts Stage or Task instances in Available state.</td>
</tr>
<tr>
<td>milestone</td>
<td>Represents an achievable target, defined to enable evaluation of progress of a Case.</td>
</tr>
<tr>
<td>onPart</td>
<td>Addresses the &quot;event&quot; aspect of a Sentry.</td>
</tr>
<tr>
<td>parameterMapping</td>
<td>Used for input/output mapping of CaseTasks and ProcessTasks.</td>
</tr>
<tr>
<td>planFragment</td>
<td>Root element for PlanItems, possibly dependent on each other, that should go into the plan as a unit. E.g. a combination of two Tasks, whereby, the completion of one Task satisfies the Sentry that enables the start of the other or a combination of an EventListener and a Task, whereby the occurrence of the event satisfies the Sentry that enables the start of the Task.</td>
</tr>
<tr>
<td>planItem</td>
<td>A planItem instance is a use of a PlanItemDefinition element in a PlanFragment (or Stage).</td>
</tr>
<tr>
<td>Case Management Model element</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>The same PlanItemDefinition might be (re-)used multiple times as part of different combinations, i.e. as part of different PlanFragments (or Stages). Hence, a PlanItemDefinition (e.g. a Task or EventListener) is defined once, and can be (re-)used in multiple PlanFragments (or Stages). This required a separate class, PlanItem, that refers to PlanItemDefinition. Multiple PlanItems might refer to the same PlanItemDefinition. A PlanItemDefinition is (re-)used in multiple PlanFragments (or Stages) when these PlanFragments (or Stages) contain PlanItems that refer to or (“use”) that same PlanItemDefinition.</td>
<td></td>
</tr>
<tr>
<td><strong>planItemControl</strong></td>
<td>Defines aspects of control of instances of Tasks, Stages, EventListeners, and Milestones. They are defined in relation to their “origins” in the model – PlanItems and DiscretionaryItems – and may be defaulted.</td>
</tr>
<tr>
<td><strong>planItemDefinition</strong></td>
<td>Is an abstract class. Defines the building blocks from which case plans are construed. It is specialised into several concepts like PlanFragment, Task, EventListener, and MileStone.</td>
</tr>
<tr>
<td><strong>planItemOnPart</strong></td>
<td>Inherits from OnPart.</td>
</tr>
<tr>
<td><strong>planningTable</strong></td>
<td>Defines the scope of planning at run-time, in terms of a sub-set of PlanItemDefinitions that can be considered for planning in a certain context. The context for planning might be (1) a Stage (including Stages and Tasks) or (2) a HumanTask. Both Role authorisations and ApplicabilityRules can dynamically control what DiscretionaryItems, possibly organised via sub-PlanningTables, are exposed to Case workers that are involved in planning.</td>
</tr>
<tr>
<td><strong>process</strong></td>
<td>Abstraction of Processes as they are specified in various Process modeling specifications.</td>
</tr>
<tr>
<td><strong>processParameter</strong></td>
<td>Parameter of a Process</td>
</tr>
<tr>
<td><strong>processTask</strong></td>
<td>Can be used in the Case to call a Business Process.</td>
</tr>
<tr>
<td><strong>repetitionRule</strong></td>
<td>Specifies under which conditions Tasks, Stages, and Milestones will have repetitions. Each repetition is a new instance of it. The trigger for the repetition is a Sentry, that is referenced as entry criterion, being satisfied, whereby an OnPart of that Sentry occurs. For example: A Task might be repeated each time a certain document is created. The Task (as PlanItem) might have an entry criterion, referring to a Sentry, having on OnPart whereby the OnPart refers to the CaseFileItem that represents the type of document, and whereby the standardEvent of the OnPart is specified as “create”. When the RepetitionRule as contained in the PlanItemControl of the Task (as PlanItem) also evaluates to “true”, the Task is repeated upon creation of the document. This rule MUST be evaluated when the Milestone, Stage or Task instance is instantiated and transitions to the Available state, and their Boolean value SHOULD be maintained for the rest of the life of the Milestone, Stage or Task instance. Stage and Task instances with a RepetitionRule evaluating to “true” will create an instance every time an entry criterion with an onPart is satisfied.</td>
</tr>
<tr>
<td>Case Management Model element</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>satisfied. Under that condition a new instance is created and because the entry criteria is satisfied it moves from the Available state to either Active or Enabled state depending on the ManualActivationRule. EventListeners cannot have RepetitionRule. However, for a TimerEventListener repetition can be defined via a timerExpression based on ISO-8601, by defining repeating intervals in it using “R&lt;n&gt;/” notation.</td>
<td></td>
</tr>
<tr>
<td>requiredRule</td>
<td>Specifies under which conditions Tasks, Stages, and Milestones will be &quot;required&quot; to complete or terminate before their containing Stage can complete. The RequiredRule determines whether the Milestone, Stage or Task instance having this condition MUST be in the Completed, Terminated, Failed or Disabled state in order for its parent Stage instance to transition into the Completed state. This rule MUST be evaluated when the Milestone, Stage or Task instance is instantiated and transitions to the Available state, and their Boolean value SHOULD be maintained for the rest of the life of the Milestone, Stage or Task instance. If this rule is not present, then it is considered &quot;false&quot;. If this rule evaluates to &quot;true&quot;, the parent Stage instance MUST NOT transition to Complete state unless this Milestone, Stage or Task instance is in the Completed, Terminated, Failed or Disabled state. This rule impacts Stage instances in Available state.</td>
</tr>
<tr>
<td>sentry</td>
<td>&quot;Watches out&quot; for important situations to occur (or &quot;events&quot;), which influence the further proceedings of a Case. It is a combination of an &quot;event and/or condition&quot;. A Sentry make take one of the following forms: 1) on &lt;event&gt; if &lt;condition&gt; 2) on &lt;event&gt; 3) if &lt;condition&gt;</td>
</tr>
<tr>
<td>A Sentry may consist of two parts: - zero or more OnParts specifying the event that serves as a trigger. When event is catched, the OnPart is said to &quot;occur&quot;. - zero or more IfParts specifying a condition as Expression that evaluates over de CaseFile. A Sentry MUST have an IfPart or at least one OnPart. If all OnParts of a Sentry have occurred, and its IfParts (if existent) evaluates to &quot;true&quot;, the Sentry is said to be &quot;satisfied&quot;. A Sentry that is satisfied actually triggers the PlanItem that refers to it.</td>
<td></td>
</tr>
<tr>
<td>stage</td>
<td>Stages may be considered “episodes” of a Case, though Case models allow for defining Stages that can be planned in parallel also. Unlike PlanFragments (that are not Stages), Stages do have run-time representations in a Case (instance) plan. 1) Is a PlanItemDefinition as well. Can be used as PlanItem inside Planfragments or other stages.</td>
</tr>
<tr>
<td>Case Management Model element</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>2) Represents a Stage in the Case Model and comprises of zero or one PlanningTable. A Stage (instance) can serve as context for planning to support users in planning additional (&quot;discretionary&quot;) items into instances of the Stage in run-time.</td>
<td></td>
</tr>
<tr>
<td>3) The Case refers to a Stage as its casePlanModel. This defines the &quot;most outer&quot; Stage of the Case. This &quot;most outer&quot; Stage also contains the PlanItemDefinitions that are used in the Case. This &quot;most outer&quot; Stage of the Case may also contain Sentries that serve as exit criteria for that Stage, and hence for the Case.</td>
<td></td>
</tr>
<tr>
<td>tableItem</td>
<td>Might be a DiscretionaryItem or a PlanningTable</td>
</tr>
<tr>
<td>timerEventListener</td>
<td>Instances are used to catch predefined elapses of time.</td>
</tr>
</tbody>
</table>
Appendix E. Process model search process

This appendix defines the search process, which takes place at the sample patent office after a patent application has been filed.

In this context one of the most important business artifacts is the patent application. A patent application is a logical set of data which the patent office is receiving from its IP (Intellectual Property) ‘customers’ (applicants) who want to protect an invention, which ‘starts its life’ at the patent as soon as it is filed.

The process model is presented in Figure E-1, Figure E-2 and Figure E-3 ‘Process model search process’. Figure E-1, Figure E-2 and Figure E-3 should read as one process model from left to right.

The activities in the process model have different colours. Each colour presents the role executing the activity. See Figure E-4 Roles executing the activities.

Roughly the search process can be broken down into the following sub processes, as presented in the process model:

- Pre-classification: a classification is given for allocating the patent application to the right patent offices (search) directorate (Figure E-1),

- Formalities checking: It is checked if all information available in the patent application, are names and addresses correct, financial validation etc. (Figure E-1),

- Distribution: allocation of the patent allocation to a patent offices (search) directorate (Figure E-1),

- Division allocation: the patent application should be technically accepted within the patent office (search directorate) and allocated to the proper person(s) within a patent offices directorate who will perform the search etc. (Figure E-1 and Figure E-2),

- Search validation: validation of the claims against the rules as defined in a convention and if applicable sending of requests for clarification (Figure E-2 from activity Rule 63(1) onwards),

- Search: a search strategy is defined and implemented. Previously published technical disclosures are identified that will be relevant for assessing the patentability of the invention described in the patent application. The patent application is examined and it is checked if the invention as claimed, is novel and involves an inventive step. These requirements are mandatory for the grant of a patent (Figure E-3),

- Search report creation: the patent examiner prepares a search report including all his/her findings, which his published and send to the applicant (Figure E-3).

When the applicant has prepared an application and there are no objections or requests for clarification from the patent examiner, a happy flow is applicable which is represented by the activities on the more or less horizontal straight line in the middle of the process model. In that situation no further involvement or communication with the applicant is required and a search report is prepared at the end of the search process.
Figure E-1 Process model search process part 1

Figure E-2 Process model search process part 2
Figure E-3 Process model search process part 3

<table>
<thead>
<tr>
<th>Colour of activity</th>
<th>Role executing the activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Formalities officer or case file manager</td>
</tr>
<tr>
<td>Red</td>
<td>Patent examiner</td>
</tr>
<tr>
<td>Green</td>
<td>Applicant</td>
</tr>
</tbody>
</table>

Figure E-4 Roles executing the activities

In sub process ‘Search validation’, the content for each of the information items of the patent application should comply with the articles and rules as formulated in a convention, as listed below at the end of this appendix.

If for example, the patent examiner considers that the patent application does not comply with the requirement of unity of invention, it shall draw up a partial search report on those parts of the application which relate to the invention, or the group of inventions within the meaning of Article 82, first mentioned in the claims. It shall inform the applicant that, for the search report to cover the other inventions, a further search fee must be paid, in respect of each invention involved, within a period of two months. The search report shall be drawn up for the parts of the application relating to inventions in respect of which search fees have been paid.

The business artefact (passive resource) Application, which is going through the whole process, consists of at least the following information items:

- patent application identification,
- date of filing,
- priority (application),
- priority date,
- classification(s) (of different type),
- description,
The information items ‘description’, ‘claims’ and ‘drawings’ are most important for the examiner to judge novelty. These information items are also part of the publication of the patent application.
Appendix F. YAWL notation

This appendix gives a brief description of the YAWL notation which can be used for process modelling and the specification of Petri-nets.

The different symbols and the meaning of each symbol are described in Figure F-1 YAWL process modelling notation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming one or OR-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing one or OR-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming one or OR-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing AND-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming one or OR-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing XOR-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming AND-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing one or OR-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming XOR-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing one or OR-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming AND-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing AND-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming XOR-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing XOR-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming AND-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing XOR-split</td>
</tr>
<tr>
<td>Activity 1</td>
<td>An activity with:</td>
</tr>
<tr>
<td></td>
<td>• Incoming XOR-join</td>
</tr>
<tr>
<td></td>
<td>• Outgoing AND-split</td>
</tr>
<tr>
<td>st-1</td>
<td>A state</td>
</tr>
<tr>
<td></td>
<td>A connection between:</td>
</tr>
<tr>
<td></td>
<td>• a state and an activity or</td>
</tr>
<tr>
<td></td>
<td>• an activity and a state</td>
</tr>
</tbody>
</table>

Figure F-1 YAWL process modelling notation

The rectangle box presents an activity.

The circle presents a process state.

The arrow presents the existence of a logical sequential connection between a process state and an activity or between an activity and a process state.

Different activities can be presented using the existence of triangles in the box.
The first box without triangle can be used for presenting an activity which results from either a single process state or several process states using an OR-join and which results in either a single process state or several process states using an OR-split.

The third box with triangle forward can be used for presenting an activity which can result from either a single process state or several process states using an OR-join and which results in several process states using an XOR-split.

When the process model shows parallel-paths because of an AND-split, the system keeps track of the parallel traces and ensures that parallel activities are initiated. When these parallel activities meet at an AND join, the system only takes the case onward, if all preceding activities are ready.

At an XOR-split the business process model, the system makes sure that only one path is chosen, and that other paths are excluded.

A resource is required to execute an activity. Some resources are executing activities (so-called active resources) and some resources undergo this execution of work (so-called passive resources).

The active resources can be human resources (like applicants, patent examiners etc.) or technical systems specifically machines or information systems. Human resources can be linked to positions in the organisational model, which are called roles. Different human resources can fulfil a role.

The passive resources are physical or informational inputs and/or outputs of an activity. A passive resource can be something required as input for an activity, can start to exist as a result of an activity or be transformed during the execution of the activity. By linking the states of passive resources, a life cycle of the passive resource can be constructed.

The notation for a passive resource life cycle consists of a circle (which presents the state of a passive resource) and an arrow between 2 circles (which presents the transition of one state of a passive resource to another).