Abstract

The main goal of this research was to improve the modeling environment of the current Case Management solution at Cordys to fully support an agile software development methodology like Scrum. Scrum is being used at the Product Division at Cordys where this research was being performed. This research originated from a bigger ongoing project at Cordys, the automation of the entire Product Division at Cordys (where Scrum and Case Management are a small part of) which is based on business innovation in general and product innovation in particular.

A literature study was performed in order to get a better understanding of Scrum and Case Management (and its current implementation at Cordys). Next, requirements were set how to refine the modeling environment of the current Case Management solution to fully support Scrum. Consensus was build around these requirements, and the prioritization of them, with several employees of Cordys Netherlands and India.

Along the road OMG initiated a request for proposal (RFP) to propose semantics (meta-model, logic and notation) for a standard of Case Management. This RFP could strengthen this research and make it more interesting and ambitious. So it was decided to join this research and the RFP initiated by OMG. Because this RFP joined this research it was decided not to extend the current Case Management solution of Cordys, but to design whole new semantics.

During some period the first supervisor at Cordys came along to help design these semantics for the modeling environment of a standard for Case Management. After designing the proposed semantics, they needed to be validated. This was done by mapping the Scrum requirements on the proposed semantics, almost all requirements could be fully mapped on the semantics. The requirements which couldn’t be fully mapped on the semantics were based on the run-time of Case Management. So there should be some further studies initiated on this subject.

Next to the mapping of the Scrum requirements on the proposed semantics, there were also some new Scrum case models designed (with the proposed notation) to express a Scrum project. This was to validate the proposed semantics, but also to keep track of the original subject of this research: Case Management to fully support a Scrum way of working.

After these semantics were validated, a discussion was started with potential OMG co-submitters. Because OMG has several (developing) members (next to Cordys), it would take too much time to reach total consensus about the proposed semantics. So this is not part of this research.

Because the RFP initiated by OMG, the suggested semantics in this research should be seen as a proposal for a standard for Case Management. Cordys can use these semantics for some incremental extensions on their current Case Management solution to fully support Scrum.
# Table of Contents

Abstract .................................................................................................................... 2  
Table of Contents ....................................................................................................... 3  
List of Figures ........................................................................................................... 6  
List of Tables ............................................................................................................ 8  
Preface ....................................................................................................................... 9  
1. Introduction ......................................................................................................... 10  
2. Innovation and this thesis .................................................................................. 11  
2.1 The need to innovate ....................................................................................... 11  
2.1.1 What is Business Innovation ..................................................................... 11  
2.1.2 Product Innovation as part of business innovation ................................. 11  
2.2 Managing Innovation ...................................................................................... 12  
2.3 Framework to Manage Innovation ................................................................ 13  
2.3.1 Quick overview of the Innovation Management Framework .............. 14  
2.4 Product Innovation Ecology .......................................................................... 14  
2.5 Placing this thesis in the framework and ecology ........................................... 15  
3. Scrum and Case Management .......................................................................... 17  
3.1 What is Scrum .................................................................................................. 17  
3.1.1 History of Scrum ....................................................................................... 17  
3.1.2 How Scrum works in practice ................................................................. 17  
3.1.3 Scrum, not only just for software development ...................................... 20  
3.2 What is Case Management and the Cordys implementation ................. 21  
3.2.1 Introduction into Case Management ....................................................... 21  
3.2.2 Cordys implementation of Case Management ....................................... 22  
3.2.2.1 As-Is Meta-model of Cordys Case Management .............................. 22  
3.2.2.2 As-Is Logic ....................................................................................... 23  
3.2.2.3 Snippet of Run-time As-Is (Executable Activities) ......................... 25  
3.2.3 As-Is Notation ......................................................................................... 26  
3.2.3.1 Explanation of the example Case Model ....................................... 26  
3.2.3.2 Mapping to ORM Meta-model ...................................................... 27  
3.2.3.3 Mapping to ORC Logic Rules ...................................................... 29  
3.2.4 Example in Run-Time (current implementation of Cordys Case Management) .......................................................... 29  
3.3 Case Management to support Scrum .............................................................. 31  
3.3.1 Must haves ................................................................................................ 31  
3.3.2 Should haves ............................................................................................ 32
3.3.3 Could haves ................................................................. 33

4. Formulating the new Formalism ........................................ 34

4.1 Request for proposal initiated by OMG .......................... 34

4.2 Concrete suggestions for new semantics .......................... 34

4.2.1 Meta-model ............................................................ 34

4.2.1.1 Case File ............................................................. 35

4.2.1.2 Case life Cycle ................................................... 36

4.2.1.3 Case Objectives .................................................. 37

4.2.1.4 Case Work Product .............................................. 37

4.2.1.5 Case Organization ................................................. 37

4.2.2 Logic ......................................................................... 37

4.2.3 Notation ................................................................. 38

5. The Modeling Process ..................................................... 44

5.1 Mapping of the requirements on the new Formalism ........ 44

5.1.1 Must haves ............................................................ 44

5.1.2 Should haves .......................................................... 53

5.1.3 Could haves .......................................................... 57

5.2 Expressing a Scrum project with the suggested notation ..... 63

5.2.1 Analysis and Design Case Model................................. 64

5.2.2 Implementation Project Case Model ........................... 65

6. Next steps in Case Management ....................................... 70

6.1 Next steps for completing Scrum support ......................... 70

6.1.1 Dynamics in case task UI ......................................... 71

6.1.2 Task templates and open with .................................. 71

6.1.3 Interactive Gantt chart ............................................. 71

6.1.4 Functional modeling of Case File ................................ 72

6.1.5 Abstracting data from Case File ................................. 72

6.2 Next steps for completing standard of Case Management ..... 72

7. Conclusion .................................................................... 74

8. References ..................................................................... 75

Appendix A: Current meta-model based on UML state machines 77

Appendix B: Meta-model current implementation Case Management of Cordys ......................................................... 78

Appendix C: Proposed ORM meta-models ............................. 79

(1) Case Root Model .......................................................... 79

(2) Case File ...................................................................... 80

(3) Case Life Cycle ........................................................... 81

(4) Event .......................................................................... 82

(5) Event Composition ....................................................... 83
<table>
<thead>
<tr>
<th>Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(6)</td>
<td>Case Work Planning (Activity Overview)</td>
<td>84</td>
</tr>
<tr>
<td>(7)</td>
<td>Case Decisions</td>
<td>85</td>
</tr>
<tr>
<td>(8)</td>
<td>Case Objectives</td>
<td>86</td>
</tr>
<tr>
<td>(9)</td>
<td>Case Life Cycle and Case Work Planning Combined.</td>
<td>86</td>
</tr>
<tr>
<td>(10)</td>
<td>Case Decision to Fragment Constraints</td>
<td>87</td>
</tr>
<tr>
<td>(11)</td>
<td>Activity Planning Decision to Case Task Constraints</td>
<td>87</td>
</tr>
<tr>
<td>(12)</td>
<td>Case Tasks</td>
<td>88</td>
</tr>
<tr>
<td>(13)</td>
<td>Case Work Product</td>
<td>88</td>
</tr>
<tr>
<td>(14)</td>
<td>Case Work Planning - Constrainable Sequence</td>
<td>89</td>
</tr>
<tr>
<td>(15)</td>
<td>Case Organization</td>
<td>90</td>
</tr>
<tr>
<td>Appendix D: Subtype defining rules for proposed meta-models</td>
<td>91</td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: Innovation radar (source: [SAW06]) .................................................. 12
Figure 2: Levels of business and product innovation (source: [SWE10]) ................. 13
Figure 3: Innovation Management Framework (source: [SWE10]) .......................... 13
Figure 4: Product innovation ecology (source: [SWE10]) .................................... 15
Figure 5: Thesis related to product innovation ecology and the interaction ............. 16
Figure 6: Thesis related to Innovation Framework ............................................... 16
Figure 7: Sprint Cycle (source: [SUT07]) ............................................................. 18
Figure 8: Sprint Burn Down Chart (source: [SUT07]) ........................................ 19
Figure 9: General UML diagram of the technical implementation of Case Management in Cordys ................................................................. 23
Figure 10: Graphical representation of formula ..................................................... 25
Figure 11: ORM part of as-is in run-time ............................................................. 25
Figure 12: Case model for ‘as is’ notation current implementation Case Management ................................................................................ 26
Figure 13: Inbox of Cordys (as is) ......................................................................... 29
Figure 14: Plan follow-up activities in run-time (as is) .......................................... 30
Figure 15: Execution of activity (as is) ................................................................. 30
Figure 16: Case Instance Manager (as is) ............................................................ 31
Figure 17: Case File Notation (source: [SWE10]) .................................................. 39
Figure 18: Life Cycle and Event Handler Notation .............................................. 41
Figure 19: Activity Notation ............................................................................. 42
Figure 20: (3) Case Life Cycle (Snippet) ............................................................. 44
Figure 21: Transition Guard Notation ................................................................ 44
Figure 22: (6) Case Work Planning (Snippet) ..................................................... 45
Figure 23: (12) Case Tasks (Snippet) ................................................................. 45
Figure 24: Meeting Activity Notation .................................................................. 45
Figure 25: (7) Case Decisions (Snippet) ............................................................. 46
Figure 26: (11) Activity Planning Decision To Case Task Constraints (Snippet) .... 46
Figure 27: Refining Follow-Up Notation ............................................................ 46
Figure 28: (3) Case Life Cycle (Snippet) ............................................................. 47
Figure 29: (4) Event (Snippet) ........................................................................... 48
Figure 30: Time Event Notation ........................................................................ 48
Figure 31: (7) Case Decisions (Snippet) ............................................................. 49
Figure 32: (14) Case Work Planning - Constrainable Sequence (Full) .................. 49
Figure 33: Dependency Notation ...................................................................... 49
Figure 34: (2) Case File (Snippet) .............................................................. 50
Figure 35: (4) Event (Snippet) ................................................................. 50
Figure 36: (6) Case Work Planning (Activity Overview) Snippet .......... 51
Figure 37: (12) Case Tasks (Full) ........................................................... 51
Figure 38: (3) Case Life Cycle (Snippet) .................................................... 52
Figure 39: (6) Case Work Planning (Activity Overview) (Snippet) .... 52
Figure 40: (6) Case Work Planning (Activity Overview) (Snippet) .... 53
Figure 41: View Gantt Chart Activity Notation ......................................... 53
Figure 42: (6) Case Work Planning (Activity Overview) (Snippet) .... 53
Figure 43: (7) Case Decisions (Snippet) ................................................... 53
Figure 44: Follow-up to Activity Cluster Notation .................................... 54
Figure 45: (2) Case File (Full) ................................................................. 54
Figure 46: Case File Modeling Notation (source: [SWE10]) ..................... 55
Figure 47: (12) Case Tasks (Snippet) ....................................................... 55
Figure 48: Task With Case File Alignment (Source: [SWE10]) ............... 55
Figure 49: (9) Case Life Cycle and Case Work Planning Combined (Snippet) .... 56
Figure 50: Terminated When Phase Exit Notation ..................................... 56
Figure 51: (7) Case Decisions (Snippet) ................................................... 57
Figure 52: Manual Follow-Up Event Triggered Notation ......................... 57
Figure 53: (6) Case Work Planning (Activity Overview) (Snippet) ......... 57
Figure 54: Mandatory Activity Notation ................................................ 57
Figure 55: (7) Case Decisions (Snippet) ................................................... 58
Figure 56: (9) Case Life Cycle and Case Work Planning Combined (Snippet) .... 58
Figure 57: Enabling Decision On Boundary Notation ............................... 58
Figure 58: (7) Case Decisions (Snippet) ................................................... 59
Figure 59: ‘Free’ follow-up Notation ....................................................... 59
Figure 60: (6) Case Work Planning (Snippet) .......................................... 60
Figure 61: (12) Case Tasks (Snippet) ....................................................... 60
Figure 62: (6) Case Work Planning (Activity Overview) (Snippet) ........... 61
Figure 63: (10) Case Decision to Fragment Constraint (Full) ................... 61
Figure 64: Pallet and Fragment Notation .................................................. 62
Figure 65: Product Innovation Ecology Snippet (source: [SWE10]) .......... 63
Figure 66: Analysis and Design Case Model ............................................ 64
Figure 67: Implementation Project Case Model ......................................... 66
Figure 68: Gantt chart example (source: http://en.wikipedia.org/wiki/Gantt_chart) ... 71
Figure 69: Case Work Planning (Activity Overview) Snippet ................. 73
List of Tables

Table 1: Events and their handlers ................................................................. 39
Table 2: Event Handler Notation ..................................................................... 40
Table 3: Decision Notation ............................................................................ 42
Table 4: Reminder into the new notation ....................................................... 64
Preface

This thesis is result of a five month lasting research and is the final assignment of the information science education at the Radboud University of Nijmegen (RU). Because I like to get more experience with the business side of the academic world, the decision was made to contact a company to perform this master thesis. Cordys soon came to mind because the innovative way they work and their innovative work on BPM (Business Process Management).

This thesis would not been as it is now without the contribution of several persons who I like to thank for their effort and support. First of all I would like to thank my supervisors: Theo van der Weide (first supervisor at the RU), Stijn Hoppenbrouwers (second supervisor at the RU), Henk de Man (first supervisor at Cordys) and Erik van de Ven (second supervisor at Cordys).

During this research at Cordys, there were also some other people who helped me a lot by some very good discussions and explanation of several aspects of this research. For their time and effort I would like to thank: Gerwin Ligtenberg, Vincent Kok, Shiva Prasad and K V S Bhargav.

Next I would like to thank Wilfried Rijsemus for allowing me to drive with him to and from Cordys and for all things he helped me with during my research.

Last but not least I would like to thank my parents, girlfriend and friends for their support and their help with reviewing this thesis.
1. Introduction

This master thesis is the result of a research conducted to conclude the master study ‘Information Science’ at the Radboud University Nijmegen and was combined with an internship at Cordys Netherlands. The subject of this research was to propose enhancements on the modeling environment (in the form of refined semantics) of the current Case Management solution at Cordys to fully support a Scrum way of working.

Scrum is an agile software development methodology which is used at the Product Division of Cordys, where this research was being performed.

Cordys suggested the idea for this research as part of a bigger project, the automation of the core processes of their Product Division. Chapter two (Innovation and this thesis) will explain this more in detail and how this thesis relates to the Product Division automation project.

The next chapter (Scrum and Case Management) will introduce the core elements of this research. The first section will be about Scrum, a short history and how Scrum is being used in practice. The last section will be about Case Management, starting with a small introduction. The modeling environment of the current implementation of Case Management at Cordys will be shown in the next section of that chapter. The last part of that section will show the requirements of the enhancements of the modeling environment of the current Case Management implementation to fully support Scrum.

The first section of chapter four (Formulating the new Formalism) starts off how a request for proposal, initiated by OMG, crossed the path of this research and was used to enhance this study. The last section will show the semantics which were proposed.

Chapter five (The Modeling Process) shows how the proposed semantics were mapped on the Scrum requirements to be validated. The last section of that chapter will show how a Scrum process can be designed in the modeling environment of the proposed version of Case Management.

The next chapter (Next steps in Case Management) will show what other studies can be done according to this research. This study will not cover every aspect of Case Management, so there are some next steps to be taken in order to get to a full standard of Case Management. This research ends with a conclusion, a bibliography and some appendices.
2. Innovation and this thesis

The need to refine the current Case Management solution at Cordys to fully support Scrum is a part of a bigger project at Cordys. The objective of this bigger project is the automation of the core processes of the Product Division (this project is currently in the analysis phase). This research will focus on the software development process following the Scrum methodology (See 3.1 What is Scrum, for more information about Scrum), which is adopted by this department.

The primary charter of the Product Division of Cordys is (product) innovation. Therefore, the Product Division (PD) automation project focuses on business innovation in general and product innovation in particular. During this project a framework has been developed, the ‘Business Innovation Management Framework’. How can we smarter manage innovation? And more specifically, how can we better manage product innovation? With these questions in mind, Cordys came up with this framework. (More can be found in [SWE10], where chapter 10 was written by Cordys (De Man et al.) and introduces the ‘Business Innovation Management Framework’.)

This chapter will show how this thesis is related to the PD automation project and how this can be related to the framework.

2.1 The need to innovate

Pinchot and Pellman (1999, [PIN99]) describe the need to innovate as: ‘the primary source of lasting competitive advantages’ and ‘other sources of competitive advantages are only temporary’. De Man et al. (2010, [SWE10]) states innovation as ‘central to the success of a company’. When we talk about the innovation for a company, we talk about ‘business innovation’. Pinchot and Pellman (1999, [PIN99]) say this is crucial to a company because: ‘brands endure only if constant innovation ... keeps them fresh’.

2.1.1 What is Business Innovation

Sawney et al. (2006, [SAW06]) defines business innovation as: ‘the creation of substantial new value for customers and the firm by creatively changing one or more dimensions of the business system’. The term business innovation also found its way to Cordys. Cordys also knows that business innovation is key to the success of the company. According to de Man et al. (2010, [SWE10]) innovation work is extremely knowledge-intensive.

2.1.2 Product Innovation as part of business innovation

The Product Division of Cordys will be the case study for this master thesis. Where business innovation affects the entire company, we focus on product innovation as this is the core purpose of the Product Division of Cordys.

Sawney et al. (2006, [SAW06]) introduced an Innovation Radar. This radar is illustrated in the next picture.
This radar shows how product innovation is related to business innovation. Their Innovation Radar is divided into twelve dimensions, grouped into four main areas. What, Who, How and Where. So innovation can be about:

- Offerings (What): What is offered to the customer?
- Customers (Who): How can we better serve our customers?
- Processes (How): How are internal processes managed?
- Presence (Where): Where is a company present in the current market?

Where business innovation focuses on all four of the main areas, product innovation only focuses on a subset of it (i.e. Networking, Brand, Platform and Solutions).

2.2 Managing Innovation

As was mentioned before, it is very important to be able to manage a knowledge intensive process such as innovation. Accenture (2009, [ACC09]) concluded, based on a recent survey, that there is need to improve the management of innovation. They observed that:

- Despite efforts to add structure to innovation, current management practices and processes have numerous shortcomings. Fewer than half of the respondents (44%) said their company had an effective, holistic approach to new product or service development and introduction.
- To raise the return on innovation investment, companies need to treat innovation with the same discipline as other functions. The most successful companies are generating profitable revenue year after year by managing innovation as a business process.

Some reasons for a company to innovate are that (based on de Man et al. (2010, [SWE10])) the organization can resolve the following issues:

- late deliveries
- cost-full operations
- insufficient quality
- difficulty in growing organizational and process maturity
- innovation at a too low pace
- low responsiveness to trends in the market
- insufficient capability to scale
2.3 Framework to Manage Innovation

To manage innovation, a well defined framework is needed. De Man et al. (2010, [SWE10]) proposed a framework for innovation management. This paragraph will be based on the book ‘Mastering the Unpredictable’ [SWE10] (and in particular on chapter 10).

Business and product innovation involve management at three different levels. These levels are illustrated below, where the left pyramid represents the levels of business innovation and the right pyramid represents the corresponding product innovation levels.

![Figure 2: Levels of business and product innovation (source: [SWE10])](image)

These three management levels have been used as basis to construct the Business Innovation Management Framework (which spans these three levels). The next figure illustrates the main concepts or entities of the innovation framework at the three levels of innovation management.

![Figure 3: Innovation Management Framework (source: [SWE10])](image)

The innovation framework consists of entities of business innovation and product innovation. Some entities which are similar in both business and product innovation
carry a more business-oriented name. In this case, the business innovation entities are added because product innovation refers to a subset of business innovation. Entities which are specific for product innovation were also added to the overall framework (e.g. Code Merger (also known as Continuous Integration) and Story).

Besides common sense and practical business experience, De Man et al. (2010, [SWE10]) based this framework on various published management frameworks and methodologies. Important frameworks and methodologies are: The Business Motivation Model\(^1\), Lean and Six Sigma\(^2\), Scrum (more information about Scrum can be found in 3.1 What is Scrum) and EVO\(^3\).

### 2.3.1 Quick overview of the Innovation Management Framework

As discussed before, product innovation is part of business innovation. The innovation management framework is more universal. The blue boxes in the framework equally represent both business and product innovation in general. Items specific to business innovation are represented by orange boxes, product innovation items are represented by green boxes.

The innovation management framework consists of the three previously discussed layers. The first layer is the so called strategic or strategic planning layer. Here the business plans are created, objectives are set and initiatives are defined. In product innovation, like product releases, and in business innovation, like transformation initiatives, we want to prioritize these different initiatives. So there will be a lot of consensus building around that.

The next layer is the tactical layer. This layer can also be referred to as a layer of transformation- or change management. Here a release can be executed (a product release or an initiative). To execute a release in context of an initiative we start by the analysis and design of the business needs belonging to that initiative. From here the solution(s) or change requirements can be derived.

Now more atomic building blocks can be created, deriving from these requirements, such as stories. Stories are a more software oriented term and are used for e.g. basic task planning and estimation. These stories are given to an implementation team to start an implementation project to implement\(^4\) the product. Or when we put our view to the business innovation, an implementation project can be started to change the business system. After that the product or the business transition can be launched.

The last layer is the operational layer. After the launch of the product or business transition, new issues might arise. When these issues impact a bigger scope, new initiatives need to be requested.

### 2.4 Product Innovation Ecology

Because this thesis is based on product innovation (using Scrum at the Product Division of Cordys); we will focus more on this part of the innovation framework.

De Man et al. (2010, [SWE10]) stated that all concepts and related activities, which were discussed in the context of the innovation framework, can be grouped in a set of main areas. These areas can be represented by a main entity or a ‘case’. They also stated that: ‘cases can be considered undertakings, endeavors or “micro-enterprises” as well: there are goals to reach, certain means that can be

---

\(^1\) Industry-wide standard that formally defines strategic planning concepts.

\(^2\) Spanning the levels of senior, middle and front-line management, and makes use of techniques such as policy deployment.

\(^3\) Evolutionary project management. An agile project management methodology.

\(^4\) Create the code, write documentation and test the final product.
considered for use, certain rules that guide further action and decision making, and there is major room for knowledge workers to plan how to actually reach the goal'.

The innovation framework can be thought of as composed from a set of interacting cases. This resulted in a ‘Product innovation ecology’, which is illustrated below:

![Figure 4: Product innovation ecology](source: [SWE10])

This ecology shows how these cases interact with each other. Abe et al. (1998, [ABE98]) stated that emulating natural systems design is the way to go to represent interaction between, in this case, cases. Which they described as: ‘innovative, adaptive organisms’. When you see these ‘cases’ as organisms, the analogy with ecology 5 is obvious.

De Man et al. (2010, [SWE10]) gave as example to read the ecology: For instance, the case “Business Plan” is responsible for and coordinates any activity that is required to define a business plan, in concert with and continuously interacting with other cases. The case “Requirement” will carry responsibility for coordinating and tracking change from definition of it, till it has been fully implemented and administrated, and it does that concurrently with other cases, such as implementation project cases, a release case, etc.

2.5 Placing this thesis in the framework and ecology

This thesis will not be based on the full product innovation management, but will only cover a part of it. This thesis is based on Scrum and Case Management, with the Product Division of Cordys as case study. The Scrum methodology relates to almost all cases in the product innovation ecology, but its core is based on the

---

5 Study of the distributions, abundance and relations of organisms and their interactions with the environment.
‘Analysis and Design’ and ‘Implementation Project’ cases. The following figures show how this thesis is related to the product innovation ecology (the cases and the interaction between them) and the innovation framework.

Figure 5: Thesis related to product innovation ecology and the interaction

As the above figures show, is that this thesis will be based on the ‘Analysis and Design’ and ‘Implementation Project’ cases and their interaction.

Figure 6: Thesis related to Innovation Framework
3. Scrum and Case Management

This chapter describes the basic elements of this master thesis. An introduction into the key-components (Scrum and Case Management) of this research can be found in the sections 3.1 and 3.2. The last section (3.3) of this chapter will describe the needs for enhancements on the current implementation of Case Management at Cordys to fully support Scrum.

3.1 What is Scrum

The most important charter of the Cordys’ Product Division is software development. To develop correct software, which meets all the customer requirements, a good methodology is needed to manage the entire software development process. Scrum is a methodology which also support just that. Sutherland and Schwaber (2007, [SUT07]) describe Scrum as: ‘a simple framework used to organize teams and get work done more productively with higher quality.’

3.1.1 History of Scrum

The original idea of Scrum came from a 1986 study by Takeuchi and Nonaka (1986, [TAK86]). Their study concluded that small and cross-functional teams produce the best results. So a methodology based on this principle was desired. The name Scrum also originates from the study of Takeuchi and Nonaka. They compared this way of working with the ‘Scrum’ term from rugby. Scrum, as a rugby term, is a formation in which “the whole team tries to go to the distance as a unit, passing the ball back and forth”.

In October 2004 this idea was again picked up by Jeff Sutherland (2004, [SUT04]) who then officially named this new methodology to support software development ‘Scrum’ and concluded this methodology was focused on agile software development.

Sutherland and Schwaber (2007, [SUT07]) stated that an agile methodology can deliver software to customers and end users faster, better and cooler. They observed that ‘coolness’ really is a requirement by companies like Google and Yahoo. To deliver software faster, better and cooler, they designed Scrum to add energy, focus, clarity and transparency to project planning and implementation. So that it will:

- Increase speed of development
- Align individual and corporate objectives
- Create a culture driven by performance
- Support shareholder value creation
- Achieve stable and consistent communication of performance at all levels
- Enhance individual development and quality of life

3.1.2 How Scrum works in practice

Sutherland and Schwaber (2007, [SUT07]) designed Scrum to adapt to changing requirements during the development process. Scrum allows teams to prioritize customer requirements and adapt the work product in real time to customer needs. By doing this, Scrum provides what the customer wants at the time of delivery (refining customer satisfaction) while eliminating waste (work that is not highly valued by the customer).
But how does this look like in practice? The following figure shows a graphical representation of the Scrum methodology.

![Sprint Cycle](source: [SUT07])

This next section is a fully based on the research of Sutherland and Schwaber (2007, [SUT07]).

The Scrum framework has three roles, ceremonies and artifacts designed to deliver working software in, so called, ‘sprints’. These sprints usually are 30-day iterations. At Cordys these sprints take two weeks to achieve a better overview on the projects’ progress.

The roles which can be played within a Scrum project are: Product Owner (representative of the client), Scrum Master (the team leader) and Team Member. Each role comes with different responsibilities and tasks. More information about these responsibilities and tasks can be found in the research of Sutherland and Schwaber (2007, [SUT07]). The ceremonies are: Sprint Planning, Sprint Review and Daily Scrum Meeting. The artifacts are: Product Backlog, Sprint Backlog and Burn Down Chart.

When a Scrum project starts the product owner prepares a list of customer requirements prioritized by business value delivered to the customer. In order to create a realistic prioritization of the requirements, the product owner also needs help from the team. The team can provide the product owner a better insight in the cost and time to realize the developing features. This list is called the Product Backlog. This product backlog can change during the entire project.

When the product backlog is defined and prioritized, the first sprint can be launched. A sprint starts off with a ‘Sprint Planning Meeting’. During this meeting a detailed plan for the iteration is developed. A Sprint Backlog is created by pulling top items from the product backlog. The team divides these items into 1-3 work day tasks and decides how much work can be done in the current sprint based on team size, available hours and level of team productivity.
After creating the sprint backlog, the goals of the sprint is to get all the items from the sprint backlog done. This will, next to coding, also include designing, testing and documenting the realized features. The end of a sprint is not marked when all the items of the sprint backlog are realized, but when the two weeks (as it is at Cordys) are over. When at the end of the sprint there are some features which are not realized, they are transferred back to the product backlog. When there is some time left (after realizing all the features of the sprint backlog), more items are pulled from the top of the product backlog.

Every day of the sprint a ‘Daily Scrum Meeting’ is planned. During this meeting every team member informs the other team members and the Scrum master about:

- What was done during the last 24 hours
- What is to be done in the next 24 hours
- What are blocking issues/deviations

When there are some blocking issues and deviations, the team can help to resolve these. The daily Scrum meetings should not take longer than 15 minutes.

As stated before, during the sprint planning meeting specific tasks that must be completed for the sprint to be successful are identified and estimated. A tool to guide the team to successful completion of a sprint is the Burn Down Chart. This burn down chart provided the team a good overview of the progress being made during the sprints. And this can be used to predict how many sprints are necessary to complete the current project. The figure below depicts a sprint burn down chart.

![Sprint Burn Down Chart](source: [SUT07])

At the end of each sprint a ‘Sprint Review Meeting’ is planned. During the first half of this meeting, the potentially shippable code is demonstrated to the product owner. The product owner determines which items on the product backlog really have been created during the past sprint and discusses with the team and Scrum master how to best prioritize the product backlog for the next sprint. During the second half of the sprint review meeting there will be a Sprint Retrospective with only the team and the Scrum master. The way the team worked together in the sprint is being assessed and they report what things could be better in the next sprint and they take corrective actions according to these reporting’s.
3.1.3 Scrum, not only just for software development

Besides that this thesis will be of good relevance towards Cordys and their software developing customers, it will also have some relevance towards Cordys’ customers who are not specifically in the software development business. Mac Iver (2009, [MAC09]) describes that Scrum can also be used for projects other than software development. Traditional project management methods often approach all projects in the same plan driven manner. Projects fail because they do not meet the requirements (i.e. objectives) set by the customer or the overall confidence in the team's ability to deliver is lost.

Mac Iver (2009, [MAC09]) stated that: “agile methods are becoming well established to address these issues in software development efforts, but agility is not just for software”. The following definition of agility was provided: “Agility is the ability to both create and respond to change in order to profit in a turbulent business environment”.

There is no talk about software development at all in this definition, but, as Mac Iver (2009, [MAC09]) describes: “rather to addressing change (and uncertainty) in ways that bring business benefit even though the business environment is ever changing”. So why not use an agile methodology like Scrum for projects other than software development?

Sutherland and Schwaber (2007, [SUT07]) as mentioned before, designed Scrum to:

- Adapt to changing requirements during the development process;
- Allow teams to prioritize customer requirements;
- Adapt the work product in real time to customer needs

Next to these advantages, Mac Iver (2009, [MAC09]) described some other advantages of using Scrum. These advantages are listed below:

- Focus the team on the real business value;
- Clarify the work streams and the alignment of the team;
- Enforce what “done” means;
- Make progress and issues visible

So it can be concluded that Scrum can be used for projects other than software development projects as well, because the advantages listed above do not solely apply on software development projects.

Sutherland (2005, [SUT05]) described that when a company has some experience in using Scrum, it can transform a company into an organization where Scrum becomes mission critical for the entire enterprise. And not just software development! Sutherland (2005, [SUT05]) concluded that when an organization uses Scrum to manage all their projects, Scrum will:

- Increase speed of development;
- Align individual and corporate objectives;
- Create a culture driven by performance;
- Support stakeholder value creation;
- Achieve stable and consistent communication of performance at all levels;
- Enhances individual development and quality of life
3.2 What is Case Management and the Cordys implementation

The other key component of this thesis is ‘Case Management’. This section will provide a better understanding of Case Management and how this is currently implemented at Cordys.

3.2.1 Introduction into Case Management

Case Management was introduced because a new process management paradigm was needed. De Man et al. (2010, [SWE10]) described that: ‘in technology companies, such as software development companies … technology support for process management never became a mainstream initiative’. When we look at manufacturing companies, the product manufacturing process itself is often well supported by e.g. Enterprise Resource Planning (ERP) systems. The problem here is that ERP systems are not designed to manage knowledge work(ers).

Niblick (2007, [NIB07]) defines a knowledge worker as: ‘a person that creates value from a mental ability to think’. Software development is performed by knowledge workers, such as engineers and their managers. So support for a process management paradigm typically suited to manage knowledge work was needed. This process management paradigm was called: Case Management.

The Object Management Group (OMG) (2009, [OMG09]) defines Case Management as: ‘each execution of a process involves a particular situation, a case, and a desired outcome for that case’. A case involves a particular subject (e.g. person, legal action, insurance claim etc.) and actions performed related to that subject will achieve the desired result (closing the case). The determination of these actions to take in each case involves human judgment and decision-making (by a knowledge worker), according to OMG (2009, [OMG09]). As a consequence of this, activities associated with a case will not occur in a predefined sequence. So Case Management can be seen more like a collection of sequences (or traces).

The actions which can be triggered by each case include business processes that can be modeled in Business Process Modeling Notation (BPMN) (See White and Miers (2008, [WHI08])). BPMN has nowadays been adopted as a business process modeling standard. In comparison with case modeling these predefined sequences (or even better, workflows) differ generally in the fact that these models are effective for well-defined, repeatable processes (according to OMG (2009, [OMG09])).

The umbrella which covers Case Management as well as predefined workflows is Business Process Management (BPM). Smart (2008, [SMA08]) describes BPM as ‘attempts to improve processes continuously’ and a ‘process optimization process’. In the context of this master thesis BPM should be taken in a broader sense.

Silver (2009, [SIL09]) states that a case is rarely a single process in the conventional BPM sense. And that it is a collection of business processes. This shows that Case Management and business processes (as in predefines workflows) come together in BPM. McCoy (2005, [MCC05]) from Gartner described BPM as a management practice that provides for governance of a business's process environment toward the goal of refining agility and operational performance. BPM is a structured approach employing methods, policies, metrics, management practices and

---

6 Software intended to manage all the information and functions of a business or company.
7 World's largest software consortium with an international membership of vendors, developers, and end users.
8 An information technology research and advisory firm, which is considered to be the lead company in BPM
software tools to manage and continuously optimize an organization's activities and processes.

When we look at BPM in the context of this master thesis it can be defined more as a framework (the umbrella). Silver (2009, [SIL09]) stated that Case Management can be a part of a BPM Suite (BPMS) where processes are initiated by some sort of event (something that has happened). This is also the case in Case Management. Not simply following a predefined path (as the predefined workflows), but ‘instead it advances through events’. During this study, Case Management and predefined workflow will be seen as a part of an overall framework called BPM.

3.2.2 Cordys implementation of Case Management

Cordys has developed an innovative platform, the Business Operations Platform (BOP). BOP provides tools to automate processes using BPM, which includes Case Management and predefined business process workflows and allowing them to interact and vice versa. BOP is not only a brand, but also a tentative name in the market for new emerging class of technology.

To express the semantics of the current implementation of Case Management in Cordys, we need to create an as-is scenario of three things:

- Meta-model
- Logic (rules)
- Notation

3.2.2.1 As-Is Meta-model of Cordys Case Management

As a basis for the ‘as-is meta-model’, the technical Meta-model of the current BOP-4 platform of Cordys was used. This Meta-model is fully designed in UML 2.0 to express the technical design of BOP-4. The model was derived from an up-to-date BOP-4 installation. But the Case Management solution is only a small part of the entire application. So the classes of this solution were used to create a basic Meta-model to express the technical design of the current implementation of Case Management. This general model is illustrated in the next figure.
Figure 9: General UML diagram of the technical implementation of Case Management in Cordys

Besides this general UML diagram, the Meta-model of Case Management (also a bit too technical) which is fully based on the UML 2.0 definition of a state machine was used. This diagram is illustrated in Appendix A: Current meta-model based on UML state machines. With those two UML diagrams as a basis, the modeling of the current implementation of Case Management in Cordys could start. To express this Meta-model, the Object Role Modeling (ORM) technique was used.

ORM is a modeling technique which harmonizes well with UML. Halpin (1998; [HAL98]) says this is because of: ‘since both approaches provide direct support for roles, n-ary associations and objectified associations’. The difference between ORM and UML is that, according to Halpin (1998; [HAL98]): ‘ORM pictures the world simply in terms of objects (entities or values) that play roles (parts in relationships)’. Because of this simplicity and the means to assign names to roles which an object can play, such a diagram can be read very easy. A more detailed overview of ORM can be found in [HAL96] and [HAL982].

Since this thesis concentrates itself fully on the modeling environment of Case Management, the ORM model of the as-is scenario is based on the modeling environment of the current Case Management solution in Cordys. The ORM Meta-model can be found in Appendix B: Meta-model current implementation Case Management at Cordys.

3.2.2.2 As-Is Logic
An ORM meta-model of the current Case Management solution at Cordys does not fully cover everything. There are certain rules and logic which cannot fully be expressed in ORM. A language to express these rules and logic which goes hand-in-
hand with ORM is Object Role Calculus (ORC). Van Bommel et al. (2006; [BOM06]) describe ORC as a variant of Lisa-D\(^9\). They suggested that Lisa-D was originally designed to describe all computable sets of facts that can be derived from the elementary facts defined in the underlying conceptual schema. Van Bommel et al (2006; [BOM06]) defines ORC as: ‘conceptual language in which rules can be expressed for describing the behavior that may be observed in a logbook compatible with the domain being modeled’

The rules and logic of the as-is scenario, which were expressed by the use of ORC, can be found on the ORM Meta-model in Appendix B. When we look at those ORC rules, we can see they are all derived from the objects and the roles they play in the ORM Meta-model. By providing well defined names to those roles, ORC is easier to read and understand. Listed below are the definitions of these rules/logic:

- **Initial State = Vertex But Not Incoming Transition**: An Initial State is a pseudo state which is a starting point which points to the first step. So an initial state has never an incoming transition. (and you can never be in a pseudo state)
- **Final State = (Vertex But Not Outgoing Transition) And (State But Not (Constrains Activity Or Sub State of Or Has sub State Or Has State Entry))**: A Final State is an end state of the lifecycle definition of a case model. Different from the Initial State, is that an instance of a case can be in a Final state. So it is a sub-object of a State which is a sub-object of a Vertex. Because it is the end state, it has no outgoing transition (which applies to Vertex). A Final State is only a state and can never constrain any activities and can never have any sub-states within it nor be a sub-state of another state. Because a Final State cannot constrain any activities, it can also never have a state entry. These last rules only apply to State.

In the ORC rule for Final State, there is a part where Or operators are used instead of And operators (State But Not (Constrains Activity Or Sub State of Or Has sub State Or Has State Entry)). Where And operators might read more easily, this decision was made when proving this with de Morgan’s law:

\[ \neg (P \lor Q) = \neg P \land \neg Q \]
\[ \neg (P \land Q) = \neg P \lor \neg Q \]

When simplifying the ORC rule and translating each role (e.g. Outgoing Transition) to a predicate and each ORC operator to a logical operator (as described by van Benthem et al. (2003; [BEN03])), we get a formula like:

\[ \neg A \land \neg (B \lor C \lor D \lor E) \]

Where \( A \) = ‘Outgoing Transition’, \( B \) = ‘Constrains Activity’, \( C \) = ‘Sub State’, \( D \) = ‘Has sub State’, \( E \) = ‘Has State Entry’. There is an \( \land \) (AND) between them, because \( A \) (Outgoing Transition) is about Vertex and \( B, C, D \) and \( E \) are about State.

This can be reformulated with de Morgan’s law as (where \( P = A \) and \( Q = B \lor C \lor D \lor E \)):

\[ \neg P \land \neg Q = \neg (P \lor Q) = \neg (A \lor B \lor C \lor D \lor E) \]

This rule will only hold when all the predicates are false. This can be expressed graphically when depicting a ‘universe’ as illustrated below. Where the predicates (\( A, B, C, D \) and \( E \)) are shown as white ovals and the union (\( A \lor B \lor C \lor D \lor E \)) stands for the rule \( A \lor B \lor C \lor D \lor E \) (the whole white part). But we are not interested when these predicates are true, but when the entire union is false. So the grey part

\(^9\) Conceptual rule language to formally reason about domains.
in the next figure, where the grey part stands for $\neg(A \cup B \cup C \cup D \cup E)$ or (when translating this to logical operators) $\neg (A \lor B \lor C \lor D \lor E)$. It was proven before (with de Morgan’s law) that this formula is logically equivalent to the ORC rule.

![Figure 10: Graphical representation of formula](image)

**3.2.2.3 Snippet of Run-time As-is (Executable Activities)**

Next to rules in the modeling environment, is there also a rule which applies in run-time. When is an activity executable? An activity is (in the current implementation of Case Management in Cordys) executable when:

- An activity is free (can be planned ad-hoc) and is not constrained by a state.
- An activity is free and the current state of the state machine is the same as the state which constrains that activity.
- An activity is follow-up alternative of a follow-up and is not constrained by a state.
- An activity is follow-up alternative of a follow-up and the current state of the state machine is the same as the state which constrains that activity.

The ORM fragment that defines that an activity is executable is illustrated below:

![Figure 11: ORM part of as-is in run-time](image)

The ORC rule which applies to if an activity is executable in run-time is (this rule applies to the ORM part of activity in run-time and ORM part of activity in modeling environment):

Let Activity Is Executable Be
(Activity Is Free And Not Constrained by State) Or Otherwise
(Activity Is Free And Constrained by State Being Current State of State Machine) Or Otherwise
(Activity Is Alternative of Follow-up And Not Constrained by State) Or Otherwise
(Activity Is Alternative of Follow-up And Constrained by State Being Current State of State Machine)
3.2.3 As-Is Notation

The last part of the semantics which will be described, will be the notation used in the current implementation of the Case Management solution in Cordys. The as is notation will be described by means of a case model example. This case model contains all the tools/notation of Case Management.

The example case is of the handling of an insurance company for cars. Let’s say you are driving to your holiday destination in Italy, but somewhere in Germany your car breaks down. You call the company where you have insured your car. Your claim will be the trigger which starts an instance of the Case Management process. The design of such a (simplified) case model is illustrated below.

![Case Model Example](image)

The illustrated case model is a graphical representation of the notation in the current implementation of Case Management in Cordys. To describe the notation, the case model will be explained and mapped to the ORM model and ORC rules.

3.2.3.1 Explanation of the example Case Model

The case model, illustrated in Figure 10, is a model which illustrates the lifecycle of a case or (as mentioned above) a collection of traces which can be followed. As stated before, a fictive car insurance company was chosen as an example. To start a case model, an instance of a case must be created. This can because someone calls the company to file a claim. This claim will be the case which will be managed. We start the case model with an ‘Initial state’. This is represented by a gray circle. From this initial (or pseudo) state there is a ‘Transition’ to a ‘State’ (Open). So the claim (instance of a case) will be in the state ‘Open’ (which will be the current status for that case). In this state we have a ‘State Entry’. This is represented by a circle containing a green arrow entering a green square.

After that, an automatic follow-up (represented by an arrow with a small circle containing an ‘A’), which automatically plans the first ‘Activity’. This first activity ‘Complete data’ can then be executed (See 3.2.4 for more information about the current implementation of Case Management in run-time). After finishing this activity the ‘Check payment’ activity will be planned automatically. But there is also a manual follow-up possible to the activity ‘Take care of salvation’. This is
represented by an arrow with a small circle containing an ‘M’. A manual follow-up means that one can choose to plan this next activity. So this activity can be planned manual. This activity is in the activity cluster ‘Check Policy’. This is represented by a dashed line covering two activities.

Next to a manual follow-up one can also plan ‘free’ activities. These are activities which can be planned ad-hoc. Activities can be in- or outside a. A free activity has no incoming follow-up arrows. Examples of free activities inside a state are: ‘Take care of transport’, ‘Complete arrangements’ and ‘Call Hotel’. There are also two free activities outside any state, these are: ‘Send SMS to customer’ and ‘Call customer for details’. The difference between activities inside or outside a state is that an activity constrained by a state can only be executed while the current state of the case is that particular state.

Next to manual and automatic follow-ups, there is also an ‘Intermediate’ follow-up. This is represented by an arrow with a small circle containing an ‘I’. This is illustrated in the example case model between the activities ‘Take case of transport’ and ‘Call finance’. An intermediate follow-up is while busy with an activity, you need to do another activity to complete the current one. An intermediate follow-up will pause the current activity so you can work on another activity. When the other activity is completed, you will return to the paused activity.

When there is a case data event like ‘Document received’, this can trigger follow-up planning. This ‘document received’ event is represented by a circle containing a sheet of paper. In the example case model, this event has an automatic follow-up which plans the activity ‘Arrange other hotel’. It could be possible that a hotel send an e-mail that the booking wasn’t successful so another hotel is needed.

There is another free activity within the state Open; this is ‘Complete arrangements’. Next to completing an activity, an activity can also raise another event. This event can be used to trigger a transition to another state. In this example the ‘Complete arrangements’ activity can raise the event ‘Complete’. Current there is no notation which graphically represents this. But a transition can be given a name. In this case the transition to the state ‘Close’ is called ‘Complete’. So that it shows which event should be raised to trigger this transition. The same thing applies to the transition back to the state ‘Open’ from the state ‘Close’. An activity inside the state ‘Close’ can raise the event ‘Re-open’. In this example this is the activity ‘Arrange new help’.

While being in the state ‘Close’, it is also possible to raise another event. The event called ‘Close’. This event triggers the transition to the ‘Final state’. The final state is really the final state of the case model. When an instance is in this state, it is the end of its life cycle. The final state is represented by a circle containing a grey circle. In this case the activity ‘Close activities’ can raise the event ‘Close’.

3.2.3.2 Mapping to ORM Meta-model

The example case model and its explanation can also be linked to the ORM Meta-model (see Appendix B), to validate the ORM model. We want to prove that the first schema (the example) is a sub-set (or can be created by) or the second schema (the ORM meta-model). So: \( \Sigma_1 \subseteq \Sigma_2 \), where \( \Sigma_1 \) is the example and \( \Sigma_2 \) is the ORM meta-model. To prove this, the example (i.e. a trace in the example) will be mapped to the ORM Meta-model.

The terms used in the example can be found in the ORM Meta-model. As an example we can see that an ‘Activity Cluster’ Has ‘Activity’. By the constraints in
the ORM model we see that an ‘Activity cluster’ can have one or more activities, but that an activity belongs to zero to one activity clusters.

In the ORM Meta-model there are also a few other terms (objects in ORM) introduced. As an example we choose the object ‘Vertex’. This is a term which originates from the UML 2.0 Meta-model of state machines. It is a name which is the super object of other sub objects (Initial State and State. Where State also has a sub object called ‘Final State’.) This makes it easy to apply one role to multiple objects.

Also we can see that activities can be ‘Free’ or is alternative activity of a follow-up. Where a follow-up can be ‘Automatic’ (A), ‘Manual’ (M) or ‘Intermediate’ (I) and a follow-up should be one of those three (and cannot be more than one of those options).

When we look at a certain trace which we can pick when we look at the example case model, we should be able to make the same path in the ORM model. An example path (or trace) of the case model can be:

Initial State → Transition → State ‘Open’ → ‘State Entry’ → ‘A’ follow-up → Complete data → ‘A’ follow-up → Check Payment (in cluster Check Policy) → ‘M’ follow-up from Complete data → Take care of salvation → Document received → ‘A’ follow-up → Arrange other hotel → Complete arrangements → Raise Complete → Transition → State ‘Close’ → ‘A’ follow-up → Call customer → ‘M’ follow-up → Close activities → Raise Close → Transition → Final State

This trace is depicted in the figure below.

Now let’s try to follow (a part of) the same trace in the ORM Meta-model:

Initial State → Outgoing Transition → Transition Target State (sub object of Vertex) → State (‘Open’) Has State Entry → State Entry Raises State Entry Event (sub object of ‘Event’) → Handled by Event Handler → Has Follow-up effect → Follow-up Is Automatic → Follow-up has Activity (‘Complete data’) → Activity Raises Case Activity Event (‘Completed activity’) (sub object of Event) → etc. to Activity ‘Check Payment’ → Activity Belongs to Activity Cluster (‘Check Policy’) → etc. to Case Data Event (‘Document received’) (sub object of Event) → Handled by Transition (sub object of Event Handler) → etc. to Final State

Now we can see that a trace in the example (T₁), is (partially, because writing out the entire trace in the ORM meta-model will be too large. But it can be done) also a trace (T₂) in the ORM meta-model. To prove that Σ₁ ⊆ Σ₂ we want to prove that a trace in the example can also be traced in the ORM meta-model. That was just been proven, so T₁ = T₂.
3.2.3.3 Mapping to ORC Logic Rules

There are also a few rules to express the logic of the current implementation of Case Management in Cordys (see Appendix B). These rules in modeling environment can be seen in the example case model (Figure 12). Those two rules are: ‘Initial State’ and ‘Final State’.

An initial state can never have an incoming transition. When we look at the object ‘Initial State’ in the ORM Meta-model, we see that this object is a sub object of ‘Vertex’. This only has incoming and outgoing transitions. That is why we need a rule here to express the logic of the initial state. The ORC rule which applies to this logic is: Initial State = Vertex But Not Incoming Transition. This ORC rule simply reflexes this logic.

A final state can never have an outgoing transition. It is the end of the lifecycle. When we look at the ORM Meta-model, it is a sub object of ‘State’ and of ‘Vertex’ (through ‘State’). So the first rule we need is to express that the final state does not have an outgoing transition. Because the final state is also a sub object of state (because an instance of a case can be in a final state and state but not in an initial (pseudo) state), we need to apply more ORC rules. A state can constrain ‘Activities’, have an ‘State Entry’, can be a sub state of another ‘State’ and can have a sub ‘State’. But those roles cannot be played by the final state. So we have to extend the rules by saying that those roles cannot be played by the final state. The ORC rule which applies to this logic is: Final State = (Vertex But Not Outgoing Transition) And (State But Not (Constrains Activity Or Sub State of Or Has sub State Or Has State Entry)).

3.2.4 Example in Run-Time (current implementation of Cordys Case Management)

Now we have seen the modeling environment semantics the current implementation of Case Management at Cordys, it is time to show a little of the run-time. So it is easier to form a mental picture of the whole Case Management solution.

As an example we use the case model illustrated in Figure 12. The Case Management solution of Cordys has the option to create a dummy instance of a case. This has been done to show a bit about Case Management in run-time.

When a new instance is created, the first task will be shown in the inbox of ‘Cordys’ (the initial state has triggered the first state and in the first state has the State Entry automatically planned the first activity ‘Complete data’). This is illustrated in the figure below:

![Figure 13: Inbox of Cordys (as is)](image)

From here it is possible to start the activity and work on it. When the work has been done, it is possible to complete the activity. When an activity is completed, one can plan follow-up activities. See figure below:
From this pop-up we can see which activities are automatically planned and which manual and free (dropdown list) can be planned.

Now when we start an activity we see all the activities which can be executed in the top of the screen and we can see if we can trigger an event which can trigger a transition. This is illustrated in the picture below:

Another possibility of the current implementation of Case Management in run-time is the Case Instance Manager. The Case Instance Manager shows how many instances of a case were created and how many of those are new instances. Another option is that it shows the current state of those instances and one can see the activities which are completed or planned of a certain instance. Even the history of an instance can be viewed with this Case Instance Manager. This is illustrated in the picture below:
3.3 Case Management to support Scrum

Now we have a better understanding of Scrum and Case Management (and its current implementation at Cordys). We can define some requirements for refining the current Case Management solution at Cordys to fully support Scrum. These requirements have been prioritized according to the MoSCoW-method. There has been reached consensus about these requirements and their prioritization with several Scrum masters and product managers at Cordys Netherlands and Cordys India.

The MoSCoW-method stands (in this case) for:

- **M** (must)
  - Without this Scrum cannot be supported
- **S** (should)
  - Without this Scrum can be supported, but with work-arounds
- **C** (could)
  - This is required for all-round Case Management solution from which also SCRUM will benefit a lot

The W(ould) has not been used prioritizing these requirements.

The requirements for refining the current Case Management solution are listed below.

3.3.1 Must have

- Transition guards (Rules on state transitions (e.g. track done criteria))
- Meeting activity (Scrum meetings). When the meeting activity is planned the members of the team automatically receive a meeting request (such as in Outlook or in Google calendar). The meeting activity has framework support for checking presence/absences and capturing minutes.
- Refining follow-up: it should be possible that manual follow-up is planned during an activity, without having to suspend the activity as is required for ‘intermediate’ follow-up. Note: that this follow-up can be planned as often as needed while the activity is in progress.
- More dynamics is required on case task UI in run-time. Instead of repeatedly having to switch between task form and inbox, it is better to
e.g. automatically generate an hyperlink or activate some form control on the task UI, at the moment follow-up is planned from that activity and can be executed directly while being in that activity, based on which the user can start the planned activity directly in the context of the overall activity. This is e.g. extremely important during the various Scrum meetings.

- Time event to trigger follow-up (e.g. daily meeting)
- Project management type of dependencies and constraints. This concept replaces the original, but not yet implemented, idea of ‘before’ dependency. It is the idea to set such dependencies and constraints in run-time, at the moment new tasks are planned interactively. The Case Management model (actually the interactive planning model) supports this by new properties that can also be graphically visible. This might range from a fully specified dependency between activities and/or clusters to sort of dependency templates where the specific is defined in run-time. Note that dependencies always have to be set between instances of activities! All what is defined in the planning model only supports this setting of dependencies.
- Raise specifically defined events based on inserts, updates and deletes in case file and case file parts.
- Next to specific tasks, also support task templates. When follow-up is planned interactively, specific tasks are planned based on these templates. By the way, follow-up relationships can point to task templates. Task templates might define: task type/spec (e.g. ‘business process modeling’, ‘drafting concept node’, etc.), link to a tool (‘open with’) such as the various applications in Cordys as well as tools outside Cordys such as wiki pages/templates, power point templates etc. Task templates might further define defaults for norm times, durations, etc.
- Use an interactive Gantt chart as view on the run-time case process and its activities in the Case Instance Monitor. The Gantt chart view can also be used during interactive planning, to insert new activities (and probably dependencies). If people are authorized they can also update activities, via the Gantt chart, at any time, e.g. to update/delete activities. Updating activities includes setting a percentage complete, etc. This way some elements of a project planning tool like MS Project, becomes part of the Cordys Case Management system. The Gantt chart can further be used for setting milestones based on states and due dates in the Case Management model.
- Gantt chart can also be viewed via an activity (e.g. view Gantt chart activity).

3.3.2 Should haves

- Follow-up to an activity cluster with the following semantics: zero or one activity instances can be planned based on zero or one activities in the cluster.
- Functional modeling of case file with case file parts, aligned with ‘entity modeling’.
• Task definition aligned with case file. It should be possible for a task to define, via associations, which case file part(s) have to be produced by the task, or may be produced, or should be used, etc.
• The possibility to make distinction between activities that show as contained in a state and that should be terminated when the state is left, versus activities that are shown as contained in a state, but that can still continue when the state is left. This distinction can be shown graphically by a marker on the activity, for example an anchor symbol.
• It is better, and often required, to be able to plan follow-up interactively even when triggered by automatically received events, such as state-entry events and case data change events. The implementation of this will require creating a follow-up decision object in run-time based on the follow-up list in the Case Management model. This object has its own state (e.g. ‘created’, ‘in-progress’, ‘completed’, ‘canceled’). A pre-defined framework task might be introduced to notify persons to take the decision (who will receive this in their inbox). Follow-up list can be assigned to roles therefore.

3.3.3 Could haves

• Visualization of mandatory activities (‘must be performed’) by e.g. exclamation mark on the activity.
• Applicability rules, and maybe availability rules, which can be defines on states and visualized as tables on the state boundary.
• It is better to visualize ‘free’ follow-up explicitly, e.g. by a ‘raise hand’ marker on the activity. Note that such an activity can also be connected via explicit follow-up relationships. In that case it can be planned both on explicit follow-up and ad-hoc.
• It should be possible to abstract hours accounting/labor costing information from the case system, based on activity instance monitoring data, including data such as norm time, percentage complete, etc., as well as presence/absences data of meeting activities, etc. Via process integration this data can be automatically processed into an ERP system.
• We can generalize the use of activity clusters. E.g. two types of clusters. One type is as described above: Follow-up to an activity cluster with the following semantics: zero or one activity instances can be planned based on zero or one activities in the cluster. The other type is: plan activities in the cluster. Follow up to an activity cluster, with the following semantics: An activity instance is planned for any activity in the cluster. Typically this cluster would also contain one or more (project management type of) dependencies between activities in the cluster. This type of activity cluster might be called a ‘Plan fragment’. (Not specifically for Scrum)
4. Formulating the new Formalism

This chapter will give an overview of how the proposed semantics for a Case Management standard (and to fully support Scrum) came to be. In the first section of this chapter (4.1) there will be explained how a new project interfered with this study and how this turned out to be very useful and interesting for this thesis. The last section of this chapter (4.2) describes how the semantics came to be and will include the new meta-models, corresponding logic and notation.

4.1 Request for proposal initiated by OMG

Now there was some consensus build around the requirements and their prioritization, it was time to get from these ideas to some concrete semantics. During this study a request for proposal (RFP), initiated by OMG (2009, [OMG09]), reached Cordys. OMG described the meaning of this RFP as: “This RFP solicits proposals for a meta-model ... to support modeling of Case Management processes.”.

This RFP gave the opportunity to do more with this study than just produce a master thesis to graduate. This study was not just about refining a current implementation of Case Management (to fully support Scrum), but creating semantics for a standard of Case Management. It was decided that this was the way to go for this study, because this would give a very positive and interesting swing to it.

When the decision was made to contribute to OMG by proposing semantics for a worldwide standard for Case Management, the idea to extend the current Case Management solution of Cordys to fully support Scrum was not totally abandoned. A proof of concept was still necessary to demonstrate that the proposed semantics are feasible. The Scrum requirements (see 3.3 Case Management to support Scrum) were used to map on the proposed semantics (see 5.1 for more information and see the actual mapping) to see if they hold.

4.2 Concrete suggestions for new semantics

Now this study was provided with an opportunity to work on a standard for Case Management, the Cordys supervisor also became involved in this study. Not only as first supervisor from Cordys, but also as fellow developer of the proposed semantics. The Cordys supervisor helped with developing these semantics, because otherwise this study would have taken far too long to be done alone and it was also important that Cordys provided correct semantics to OMG (so that multiple people were involved in its creation to limit the number of errors and provide enough draft).

4.2.1 Meta-model

The first thing, when creating these semantics, was creating a meta-model. Because of the RFP initiated by OMG we decided to let the current Case Management solution at Cordys loose and start from scratch. When refining an existing meta-model, there can be some flaws in it that haven’t been noticed before or there is not enough room for refining the things that are absolutely necessary without starting all over again. So it was easier to start with a blank sheet of paper. Or in this case with a blank sheet in a modeler program.
When started designing a new meta-model, it seemed important to go back to the roots of Case Management. Decisions. Case Management is all about making decisions how to proceed. In the current Case Management solution of Cordys, there was not really such a thing. Except for planning activities after finishing another activity or when some event occurred.

Another important aspect which should be more refined in the current implementation of Case Management of Cordys was the central case file. It seemed more feasible to let a case more interact with the case file than it already did. To produce and consume case file parts. Maybe tracking absence and capturing minutes (when we look at a meeting), using a tool (excel, word, etc.) or provide the opportunity to use a template for activities and the case file part(s) they are mutating.

As the meta-model of the current implementation of Cordys (see Appendix B) was a model which fitted unto one page (including logic), soon came the realization that the new meta-model wouldn’t fit on one page. The first step was to divide a case into its main subjects. These subjects are:

- **Case File**
  - Case file records case proceedings and provides means to drive the case, through rule evaluation and event raising.

- **Case Life Cycle**
  - The modeling environment of the case (the life cycle or state machine).

- **Case Work Planning**
  - The definition of planning of the work, in the modeling environment.

- **Case Objective**
  - The objectives of a case.

- **Case Work Product**
  - The work product (physical or digital) which a case can (or must) produce.

- **Case Organization**
  - The (dynamical) organization of a case.

A top-down approach was used while designing the meta-models. When the main subjects were identified, each subject could be more detailed. The final proposed ORM meta-models can be found in Appendix C: Proposed ORM meta-models and the corresponding subtype defining rules can be found in Appendix D: Subtype defining rules for proposed meta-models. In the next sections a small overview will be provided of these detailed meta-models.

### 4.2.1.1 Case File

This meta-model gives an overview of the definitions of a case file. A case file consists of attributes, annotation and case file parts. A case file (the same as case file parts) can be associated with another case file (or part). This association can be of three different kinds:

- **Reference**
  - Just a simple reference to another case file (part).

- **Hierarchy**
  - A case file (part) is part of a (more complex) hierarchy.

- **Cross Reference**
  - E.g. a patient can have a dossier at more hospitals. This is so called cross reference.

A case file, through case file parts, can contain or refer to information that is defined through information entities and have a document attached to it.
4.2.1.2 Case life Cycle

This is the (as the name does imply) life cycle (or state machine) of a case. A Case Life cycle consists of:

- **Vertex**
  - This is a collective noun of all the states (or phases) of a life cycle. This includes normal phases (including the final state) and pseudo states (i.e. initial state).

- **Transition**
  - The transition from one vertex to another.

- **Event + handler**
  - The events which can occur during a life cycle of a case and the handlers which catch the events which are being thrown during the entire life cycle.

There are also some constraints in this model. Constraints on the transitions (start and finish) and constraints on the event handlers (e.g. when they are able to handle an event or when to trigger a follow-up action).

Next to the life cycle of a case, this meta-model has much more detail. The following meta-models are part of this bigger picture of case life cycle:

- **Event**
  - All the sort events that are possible and what objects can throw such an event.

- **Event Composition**
  - To handle more complex event (patterns). E.g.: (Event A AND Event B) OR Event C AND NOT Event D.

- **Case Work Planning (Activity Overview)**
  - This meta-model shows the different kind of activities which the proposed semantics for Case Management will provide. This will not only be tasks but also: decisions (which can now also be planned), sub processes (BPMs), sub cases (other case model), any activity (plan any activity in a state/phase) and two types of activity clusters (a cluster which holds one or more other activities). I.e. fragment (kind of an embedded sub-case) & pallet (able to plan zero or one instances from zero or one tasks in the pallet).

- **Case Decisions**
  - A more detailed view of all the different decisions which can be taken (or planned) during the life cycle of a case. These types of decisions are: enabling decisions (applicability and availability (for execution)), planning decisions (of activities), work assignment decisions (assigning activities to an assignee) and disposition decision (what to do next).

- **Case Life Cycle and Case Work Planning combined**
  - The activity execution constraint to a phase. An activity belongs to a phase (or can be in the root phase) and can only be made executable (available) when the life cycle is in that specific phase. This meta-model also provides the option to set a phase exit impact kind. With this one can set what to do with an activity when the phase it belongs to is left and the activity was still running (proceed, suspend (maybe one can return to that state later on), terminate or do nothing).

---

10 A disposition is a habit, a preparation, a state of readiness, or a tendency to act in a specified way. When a work product does not meet specifications, what to do next.

11 When an activity does not specifically belongs to a phase, it is always bound to the root phase. The blank sheet in the modeling environment modeler.
• Case Decision to Fragment constraints
  o There is the possibility to make some decision continuous while being in a phase, but a fragment (a kind of activity cluster or embedded sub case) can also hold some decisions which can be taken continuous while that planning fragment is active/running. These decisions are: enabling decisions and planning decisions.

• Activity Planning Decision to Case Task Constraints
  o It might be possible that during the execution of an activity, there is the need to plan and execute another activity. Maybe because it is part of it (e.g. execute activities during a meeting) or there is additional information needed before it is able to finish the current activity. There is also the possibility to set if the current activity should be suspended (when more information is needed) or not (e.g. when in a meeting, the meeting will continue).

• Case Tasks
  o This meta-model will provide more detail on the task activity and its relationship to the case file. Which case file parts must/can be created and which case file parts must/can be consumed (read/update).

• Case Work Planning - Constrainable Sequence
  o This meta-model shows the constrainable sequences which can be set between activities. These sequences are: Finish-Finish, Start-Finish, Finish-Start and Start-Start.

4.2.1.3 Case Objectives
This meta-model shows which objectives are set for this specific instance of a case. How these objectives can be measured and if they are constrained or not.

4.2.1.4 Case Work Product
The work products which can be created during a specific instance of a case and the possible sub work products belonging to that work product. There is also the possibility to set specs for that work product and set of which kind it is (this might differ from case to case).

4.2.1.5 Case Organization
This meta-model shows the organization belonging to that specific case. This meta-model has some other elements from other models which are currently still under development by OMG. Organization Structure Modeling and Value Delivery Modeling. For more information about these subjects see the website of OMG (http://www.omg.org).

4.2.2 Logic
Because the proposed meta-models where modeled using ORM, the decisions to use ORC again to express the logic belonging to the meta-models was easily made. The ORC logic belonging to the ORM meta-models can be found below.

(2) Case File:
• A Case File cannot be associated with itself
  o Not (Case File Associated with That Case File)
• A Case File Part cannot be associated with itself
  o Not (Case File Part Associated with That Case File Part)
(3) Case Life Cycle:
- A root phase has no transitions incoming or outgoing
  - Life Cycle Phase Root of Case Life Cycle But Not (Incoming Life Cycle Transition Or Outgoing Life Cycle Transition)
- An initial phase can have no incoming transitions
  - Let Pseudo Be Initial But Not Incoming Life Cycle Transition
- A final phase can have no outgoing transitions
  - Let Pseudo Be Final But Not Outgoing Life Cycle Transition
- A life cycle phase cannot be a sub-phase of itself
  - Not (Life Cycle Phase Sub-phase of That Life Cycle Phase)
- A root phase cannot have a Phase Entry Handler and Phase Exit Handler
  - Life Cycle Phase Root of Case Life Cycle But Not (Has Phase Entry Handler Or Has Phase Exit Handler)
- A root phase cannot be a sub-phase of another phase
  - Life Cycle Phase Root of Case Life Cycle But Not Sub-phase of Another Life Cycle Phase

(9) Case Life Cycle and Case Work Planning Combined
- A Close Case (see (6) Case Work Planning (Activity Overview)) execution can only be constrained to the Root (life cycle phase) of a Case Life Cycle (see (3) Case Life Cycle).
  - Let Close Case Be Activity Constrained to Activity Execution To Phase Constraint Belongs to Life Cycle Phase Being Root of Case Life Cycle

(13) Case Work Product
- A Case Work Product cannot be a sub-product of itself
  - Not (Case Work Product Sub-product of That Case Work Product)

4.2.3 Notation
After a basis has been laid down, it was time to think about the graphical representation of proposed semantics of Case Management. To have a theory is one thing, but is equally important to have an idea how this can work in practice. For proof of the semantics see chapter 5: The modeling process.

The first element of the meta-model was the case file. The following image provides a conceptual representation of a case file, based on the meta-model. A representation like this can also be used as basis for a graphical model notation for case file.
The next subject of the meta-model is the case life cycle. The notation of expressing a case life cycle will remain the same as it was in the current Case Management solution of Cordys (phases and their transitions) except for the graphical representation of the final and initial state. These will get the same graphical representation of the Start and End case handlers. A start and end case icon are used when a case does not have any phases (except the initial/start and final/end). These are graphically depicted by resp. a green and red dot. To make it easier for the end user, it was decided to use the same icons for start and initial and end and final.

An aspect of the case life cycle, are the corresponding events and their event handlers. To get a better understanding of which handlers can handle what kind of event, the following table was created.

<table>
<thead>
<tr>
<th>Type</th>
<th>Life Cycle Transition</th>
<th>Phase Entry Handler</th>
<th>Phase Exit Handler</th>
<th>Local Event Handler</th>
<th>Case Created Handler</th>
<th>Fragment Instantiated Handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case File Event</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disposition Event</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity Event</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase Event</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Event</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Event</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Composite Event</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Events and their handlers
The different kind of event handlers can be graphically shown like the suggestions in the table below.

<table>
<thead>
<tr>
<th>Event Handler</th>
<th>Notation</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Life Cycle Transition</td>
<td><img src="event_e4.png" alt="event e4" /></td>
<td>Handles case file event</td>
</tr>
<tr>
<td>Phase Entry Handler</td>
<td><img src="phase_entry.png" alt="phase entry" /></td>
<td>Handles disposition event, activity event or phase event (the latter only in other case); notation is implicit when the handler handles an activity completion event</td>
</tr>
<tr>
<td>Phase Exit Handler</td>
<td><img src="phase_exit.png" alt="phase exit" /></td>
<td>Handles time event</td>
</tr>
<tr>
<td>Local Event Handler</td>
<td><img src="local_event.png" alt="local event" /></td>
<td>Handles external event (or &quot;message event&quot;)</td>
</tr>
<tr>
<td>Case Created Handler</td>
<td><img src="case_created.png" alt="case created" /></td>
<td>Handles composite event</td>
</tr>
<tr>
<td>Fragment Instantiated Handler</td>
<td><img src="fragment_instantiated.png" alt="fragment instantiated" /></td>
<td>The same notation is re-used, for user convenience; we will re-use it for &quot;initial&quot; (pseudo phase) as well</td>
</tr>
</tbody>
</table>

**Table 2: Event Handler Notation**

As stated before, the case life cycle (phases and transition) notation will stay the same. The next picture will give an overview of how such a life cycle will look like with some event handlers in place.
An important aspect of the modeling environment of Case Management is the activities which can be planned and executed. The meta-model holds several different activities. The next picture will give an overview of the graphical representation of all the different activities in the proposed meta-models.

- **Task T1**
  - A normal task which can be planned and executed.
- **Task T2**
  - A normal task, but mandatory.
- **Sub-process P11**
  - A task which triggers a pre-defined workflow (in BPMN).
- **Edit Work Plan**
  - This task opens a work plan of the current case in e.g. a Gantt chart like view.
- **Sub-Case C11**
  - A task which triggers another case model.
- **Pallet P1**
  - An activity cluster which can be planned where the user can plan zero or one instances from zero or one tasks in this cluster.
- **Pallet P2**
  - The same as Pallet P1, but closed.
- **Fragment F1**
  - An activity cluster which can be seen as an embedded sub-case. This fragment starts another planning fragment next to the planning fragment of the case itself. This fragment is shown in closed state.
- **Meeting**
  - A meeting task.
- **Any**
  - ‘Any Task/Activity’. When this task is planned, the case worker can plan any activity in the current phase (or from the root phase).
- **Close Case**
  - This task closes the current instance of the case. The final state.
Next to the activities depicted above, there is also another, very important, activity. The foundation of Case Management, the decision. The following table lists the icons that are used for the various types of decisions.

<table>
<thead>
<tr>
<th>Decision</th>
<th>Notation</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Applicability Decision</td>
<td>![Activity Applicability Decision Icon]</td>
<td>Re-use Activity Applicability Decision icon, both being Activity Enabling Decisions (because they depend on another)</td>
</tr>
<tr>
<td>Activity Planning Decision</td>
<td>![Activity Planning Decision Icon]</td>
<td></td>
</tr>
<tr>
<td>Activity Availability Decision</td>
<td>![Activity Availability Decision Icon]</td>
<td></td>
</tr>
<tr>
<td>Work Assignment Decision</td>
<td>![Work Assignment Decision Icon]</td>
<td></td>
</tr>
<tr>
<td>Disposition Decision</td>
<td>![Disposition Decision Icon]</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Decision Notation

These icons listed above are just the icons of the decisions. When they are planned and executed as a task (so not continuous while in a phase) the icon can be shown on a task (quite the same as the meeting icon on a task).
Below is a description of the different kind of decisions:

- **Applicability Decision**
  - The outcome of a decision of this type is a set of zero or more activities that are determined to be "applicable". A decision of this type is normally taken automatically by the system, although humans might sometimes decide on applicability as well. Activity applicability serves as filter over activity planning decisions. Normally the system can take activity applicability decisions dynamically, based on changed or changing information in the case file.

- **Planning Decision**
  - An activity planning decision is often taken by human case workers (i.a. "manual" decision), but can be taken by the system as well (i.e. "automatic decision"). A decision maker plans zero or more activities (thereby adding copies or instances of the activities to the emerging case process or case), based on a, possibly constrained, list of activities that are eligible for planning in a certain context.

- **Availability Decision**
  - Distinction should be made between activity applicability and availability. Activities that are applicable can be planned, and activities that are planned, can be executed, but only once they are "available". It can happen that various types of constraints are defined in the case model and/or in the emerging case process itself. Only when such constraints are satisfied, the corresponding activities become ‘available’. One such type of constraints is contained in activity availability decisions. An activity availability decision determines, normally dynamically and automatically by the system, whether or not an activity can be made available for execution. Example: a certain type of document should be available in the case file before a certain activity can be executed. At the moment an assignable activity becomes available, it is inserted in the work-list of the participant to which the activity is assigned.

- **Work Assignment Decision**
  - It is important that any distinct part of - often human - activity, including the various types of decisions, are planned, monitored, audited and possibly constrained by the case management system. This includes work assignment decisions, or decisions that decide "who" should do work (assignable activity, including tasks and decisions).

- **Disposition Decision**
  - A disposition is the act of disposing, arranging, ordering, regulating, or transferring. Or: a state of readiness, or a tendency to act in a specified way. Example: in case of a non-conformance, e.g. detected during a review or inspection, possible dispositions, often predefined and formalized via specified disposition codes, are: scrap, return to vendor, repair, accept, disassemble, etc. In the context of cases, such dispositions will be tracked as case file parts in the case file, based on which new activity planning and/or case life cycle transitions can be triggered accordingly. Note that a formalized way of dealing with e.g. "abnormalities" is necessary in many case work environments.

The rest of the notation will be explained in the section 5.1: Mapping of the Requirements on the new Formalism.
5. The Modeling Process

This chapter will show how the proposed semantics were tested by a proof of concept. In the first section (5.1) it will show how the requirements (set to let Case Management fully support Scrum) are mapped to the proposed semantics. The last section (5.2) will show two Scrum case models which were creating using the proposed notation.

5.1 Mapping of the requirements on the new Formalism

When creating and eventually submitting these semantics, it is always necessary to proof that these semantics are correct. During this study this was done by a proof of concept in which the proposed semantics were test via a Scrum case study. For this Scrum case study, the requirements for the enhancements on the current Case Management solution of Cordys were used to see if they could fit within the proposed semantics. This was done by checking if these requirements were covered in the meta-model (which was the most important aspect of all the semantics), logic and notation (which are also important if Cordys wants to create an improved modeling environment for their new Case Management solution). These requirements were mapped in the tables listed below (note that all the appendices point to the subsections of Appendix C: Proposed ORM meta-models).

NB: The following sections could be used as a first step for Cordys to create some teaching method into the new formalism.

5.1.1 Must haves

<table>
<thead>
<tr>
<th>Requirement:</th>
<th>Transition guards (Rules on state transitions (e.g. ‘Track Done Criteria’))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix:</td>
<td>(3) Case Life Cycle</td>
</tr>
<tr>
<td>ORM meta-model:</td>
<td></td>
</tr>
<tr>
<td>Event Handler</td>
<td></td>
</tr>
<tr>
<td>Has / Belongs to</td>
<td></td>
</tr>
<tr>
<td>Constraint</td>
<td></td>
</tr>
<tr>
<td>Life Cycle Transition</td>
<td></td>
</tr>
<tr>
<td>Figure 20: (3) Case Life Cycle (Snippet)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORC logic:</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notation:</td>
<td></td>
</tr>
<tr>
<td>Figure 21: Transition Guard Notation</td>
<td></td>
</tr>
</tbody>
</table>
### Note:
Constraint details can be found in the properties of the guard (the diamond).

### Requirement:
Meeting activity (Scrum meetings). When the meeting activity is planned, the members of the team automatically receive a meeting request (such as in Outlook or in Google calendar). The meeting activity has framework support for checking presence/absence and capturing minutes.

### Appendix:
(6) Case Work Planning (Activity Overview)
  (Meeting activity)
(12) Case Tasks
  (Checking presence/absence, capturing minutes)

### ORM meta-model:

![Figure 22: (6) Case Work Planning (Snippet)]

### ORC logic:
- 

### Notation:
![Sprint Review Meeting]

### Figure 24: Meeting Activity Notation

### Note:
A ‘Task’ can produce a (or multiple) ‘Case Work Product(s)’ which can track a ‘Case File Part’. So absence/presence can be stored in a ‘Case File Part’. Also minutes can be captured in a ‘Case File Part’.

### Requirement:
Refining follow-up: it should be possible that a manual follow-up is planned any time during an activity, without having to suspend the activity as is required for ‘intermediate’ follow-up.

### Appendix:
(7) Case Decisions
  (Follow-up activity ‘AnyTime’)
(11) Activity Planning Decision To Case Task Constraints
  (Suspend Activity, Task constrains ‘Planning Decision’)

![Figure 23: (12) Case Tasks (Snippet)]
ORM meta-model:

![Diagram of ORM meta-model](image)

**Figure 25:** (7) Case Decisions (Snippet)

**Figure 26:** (11) Activity Planning Decision To Case Task Constraints (Snippet)

**ORC logic:**

- 

**Notation:**

![Diagram of Notation](image)

**Figure 27:** Refining Follow-Up Notation

**Note:**

The ‘I’ follow-up now has two options. To suspend ‘Task 1’ or not while planning and execution ‘Task 2’ and ‘Task 2’ must be completed before ‘Task 1’ is or not.

These options are not graphically shown in modeling environment. These options can be set in the properties of the ‘I’ follow-up.

The ‘Activity Planning Decision’ can be taken ‘AnyTime’ while executing ‘Task 1’.
### Requirement:
More dynamics is required on case task UI in run-time. Instead of repeatedly having to switch between task form and inbox, it is better to e.g. automatically generate an hyperlink or activate some form control on the task UI, at the moment follow-up is planned from that activity and can be executed directly while being in that activity. Based on which the user can start the planned activity directly in context of the overall activity. This is e.g. extremely important during various Scrum meetings.

### Appendix:
- 

### ORM meta-model:
- 

### ORC logic:
- 

### Notation:
- 

### Note:
This is behavior in run-time UI. This thesis is based on the modeling environment of Case Management, so this requirement should be handled in future studies about Case Management (see chapter ‘Future studies’).

### Requirement:
Time event to trigger follow-up (e.g. daily meeting).

### Appendix:
(3) Case Life Cycle  
   (Local Event Handler)  
(4) Event  
   (Time Event)

### ORM meta-model:
![Diagram](image)

Figure 28: (3) Case Life Cycle (Snippet)
Figure 29: (4) Event (Snippet)

| ORC logic: | - |
| Notation: | ![Image](image.png) |
| Note: | A ‘Local Event Handler’ can handle several kinds of events. One of them being the ‘Time Event’ (which can be relative or absolute). The graphical representation of a ‘Local Event Handler’ which handles a ‘Time Event’ looks like the timer icon from BPMN. |
| Requirement: | Project management type of dependencies and constraints. This concept replaces the original, but not yet implemented, idea of ‘before’ dependency. It is the idea to set such dependencies and constraints in run-time, at the moment new tasks are planned interactively. The Case Management model (actually the interactive planning model) supports this by new properties that can also be graphically visible... |
| Appendix: | (7) Case Decisions  
(constraints on Planning Decision)  
(14) Case Work Planning - Constrainable Sequence  
(dependencies and constrainable start/finish) |
ORM meta-model:

Figure 31: (7) Case Decisions (Snippet)

Figure 32: (14) Case Work Planning - Constrainable Sequence (Full)

ORC logic:

Notation:

Figure 33: Dependency Notation

Note: An activity planning decision can have multiple constraints, which can have case file elements as operands. This can be seen as a decision table. Which alternatives (follow-up activities) are plan-
able under which circumstances. Next to these plan constraints, an activity can also have a constraint on the start and finish. See ‘Moment Constraint Kind’ in meta-model.

Dependencies between activities are depicted as ‘Sequence Constraint Kind’, such as: Finish-Start, Start-Start etc.

The Constraint Kinds can be set in the properties of the activity (for start/finish moment) or the dashed arrow between activities (for sequence).

These constraints will only function as a guide, because dependencies can only be set in run-time, between instances of Tasks.

<table>
<thead>
<tr>
<th>Requirement:</th>
<th>Raise specific defined events based on inserts, updates and deletes in case file and case file parts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix:</td>
<td>(2) Case File (Case File and Case File Parts) (4) Event (Case File Events)</td>
</tr>
<tr>
<td>ORM metamodel:</td>
<td><img src="image" alt="ORM diagram" /></td>
</tr>
<tr>
<td><strong>Figure 34:</strong> (2) Case File (Snippet)</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Figure 35:</strong> (4) Event (Snippet)</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
| ORC logic: | A Case File cannot be associated with itself
- Not (Case File Associated with That Case File)
A Case File Part cannot be associated with itself
- Not (Case File Part Associated with That Case File Part) |
| Notation: | - |
| Note: | A Case File Part is part of the Case File. So when something happens in a Case File Part, the Case File will change. Then the |
Case File can raise a Case File Event. The kinds of Case File Events are depicted in ‘Case File Event Kind’.

| Requirement: | Next to specific task, also support task templates. When follow-up is planned interactively, specific tasks are planned based on these templates. By the way, follow-up relationships can point to task templates. Task templates might define: task type/spec and link to a tool. Task templates might further define defaults for norm times, durations, etc. |
| Appendix: | (6) Case Work Planning (Activity Overview)  
(Norm Time, Duration, Task Spec)  
(12) Case Tasks  
(Template, Open with tool) |
| ORM metamodel: | ![ORM Diagram](image)  
Figure 36: (6) Case Work Planning (Activity Overview) Snippet |
| ORC logic: | - |
| Notation: | - |
| Note: | These options can be set in the properties of the Task. |
 Requirement: Use an interactive Gantt chart as view on the run-time case process and its activities in the Case Instance Monitor. The Gantt chart view can also be used during interactive planning, to insert new activities (and probably dependencies). If people are authorized they can also update activities, via the Gantt chart, at any time, e.g. to update/delete activities. Updating activities includes setting percentage complete, etc.

 Appendix: (3) Case Life Cycle
(Milestone, Phase Duration, Transition Constrainable Moment)
(6) Case Work Planning (Activity Overview)
(Activity Norm Duration, Planning Percentage, Norm Time)

 ORM meta-model:

![Figure 38: (3) Case Life Cycle (Snippet)](image)

![Figure 39: (6) Case Work Planning (Activity Overview) (Snippet)](image)

 ORC logic: -

 Notation: This is a run-time view.

 Note: These settings can be changed in run-time, in the Gantt chart view.
5.1.2 Should haves

<table>
<thead>
<tr>
<th>Requirement:</th>
<th>Follow-up to an activity cluster with the following semantics: zero or one activity instances can be planned based on zero or one activities in the cluster.</th>
</tr>
</thead>
</table>
| Appendix:    | (6) Case Work Planning
(Activity Cluster)
(7) Case Decisions
(Follow-up Activity) |
| ORM meta-model: | ![Diagram of Activity Planning Decision](image) |
ORC logic: A Case File cannot be associated with itself
- Not (Case File Associated with That Case File)

A Case File Part cannot be associated with itself
- Not (Case File Part Associated with That Case File Part)
### Notation:

![Case File Modeling Notation](source: [SWE10])

### Note:
This is an entire new modeling tool. This is beyond the scope of this thesis and should be taken into further studies.

### Requirement:
Task definition aligned with case file. It should be possible for a task to define, via associations, which case file part(s) have to be produced, consumed etc.

### Appendix:

1. **Case File**
   - (see previous requirement for detailed ORM meta-model of Case File)
2. **Case Tasks**
   - (Produce and Read/Update Case File Parts)

### ORM meta-model:

![ORM meta-model](source: [SWE10])

### ORC logic:
- 

### Notation:

![Task With Case File Alignment](source: [SWE10])
### Note:
The possibility for a task to produce or read/update a case file part can be set in the properties of that task.

### Requirement:
The possibility to make distinction between activities that show as contained in a state and that should be terminated when the state is left, versus activities that are shown as contained in a state, but can still continue when the state is left.

### Appendix:
(9) Case Life Cycle and Case Work Planning Combined
(Un Plan On Phase Exit, Phase Exit Impact Kind on Activity)

### ORM meta-model:

<table>
<thead>
<tr>
<th>ORM meta-model:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Life Cycle</strong></td>
</tr>
<tr>
<td><strong>Phase</strong></td>
</tr>
<tr>
<td><strong>Activity Execution</strong></td>
</tr>
<tr>
<td><strong>To Phase Constraint</strong></td>
</tr>
<tr>
<td><strong>Activity</strong></td>
</tr>
<tr>
<td><strong>Phase Exit</strong></td>
</tr>
<tr>
<td><strong>Impact Kind</strong></td>
</tr>
<tr>
<td><strong>Has / Belongs to</strong></td>
</tr>
<tr>
<td><strong>Constrains / Constrained to</strong></td>
</tr>
<tr>
<td><strong>Belongs to / Has</strong></td>
</tr>
<tr>
<td>{Proceed, Suspend, Terminate, None}</td>
</tr>
</tbody>
</table>

**Figure 49:** (9) Case Life Cycle and Case Work Planning Combined (Snippet)

### ORC logic:
A Close Case (see (6) Case Work Planning (Activity Overview)) execution can only be constrained to the Root (life cycle phase) of a Case Life Cycle (see (3) Case Life Cycle).
- Let Close Case Be Activity Constrained to Activity Execution To Phase Constraint Belongs to Life Cycle Phase Being Root of Case Life Cycle

### Notation:

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
</tr>
</thead>
</table>

**Figure 50:** Terminated When Phase Exit Notation

### Note:
Task 1 is shown with a small red marker in the lower right corner. This shows that an activity must be terminated before the phase is left. Task 2 can continue when the phase has been left.

### Requirement:
It is better, and often required, to be able to plan follow-up interactively even when triggered by automatically received events, such as state-entry events and case data change events.

### Appendix:
(7) Case Decisions
(Manual Activity Planning Decision, Event Triggered)
ORM meta-model:

<table>
<thead>
<tr>
<th>Decision</th>
<th>Activity Decision</th>
<th>Activity Planning Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>Manual</td>
<td></td>
</tr>
<tr>
<td>Anytime</td>
<td>Event Triggered</td>
<td>When Planned</td>
</tr>
</tbody>
</table>

Figure 51: (7) Case Decisions (Snippet)

ORC logic: -

Notation:

Figure 52: Manual Follow-Up Event Triggered Notation

Note: An Activity Planning Decision can be taken ‘Manual’ when ‘Event Triggered’. This can be any event. This is shown as an ‘M’ follow-up from an Event Handler.

5.1.3 Could haves

Requirement: Visualization of mandatory activities (‘must be performed’) by e.g. exclamation mark on the activity.

Appendix: (6) Case Work Planning (Activity Overview)
(Must be performed)

ORM meta-model:

| Activity (Name) | Assignable Activity | MustBePerformed |

Figure 53: (6) Case Work Planning (Activity Overview) (Snippet)

ORC logic: -

Notation:

Figure 54: Mandatory Activity Notation

Note: -

Requirement: Applicability rules, and maybe availability rules, which can be defined on states and visualized as tables on the state boundary.

Appendix: (7) Case Decisions
(Applicability and Availability Decisions)
(9) Case Life Cycle and Case Work Planning Combined
(Enabling decisions constrained to state/phase)

ORM meta-model:

![ORM meta-model diagram]

Figure 55: (7) Case Decisions (Snippet)

ORC logic:

A Close Case (see (6) Case Work Planning (Activity Overview)) execution can only be constrained to the Root (life cycle phase) of a Case Life Cycle (see (3) Case Life Cycle).

- Let Close Case Be Activity Constrained to Activity Execution To Phase Constraint Belongs to Life Cycle Phase Being Root of Case Life Cycle

Notation:

![Notation diagram]

Figure 57: Enabling Decision On Boundary Notation

Note:

Because availability decisions also depend on applicability decisions (a Task can become not applicable anymore after it has been planned), they are shown together in one single decision table icon with a traffic light. This illustrates the applicability...
and availability decision table defined on that phase. These decision tables can be set in the properties of that decision table icon.

| Requirement: | It is better to visualize ‘free’ follow-up explicitly, e.g. by a ‘raised hand’ marker on the activity. Note that such an activity can also be connected via explicit follow-up relationships. In that case it can be planned both on explicit follow-up and ad-hoc. |
| Appendix: | (7) Decision (Any Time Planning Decision and When Planned) (9) Case Life Cycle and Case Work Planning Combined (Decision to Phase constraint. See previous requirement, same For Activity Planning Decision). |
| ORM meta-model: | ![ORM meta-model](image)  
\[Figure 58: (7) Case Decisions (Snippet)\] |
| ORC logic: | - |
| Notation: | ![Notation](image)  
\[Figure 59: ‘Free’ follow-up Notation\] |
| Note: | Task 2, 3 and 4 can be planned ad-hoc (see ‘raised hand’). The planning decision on boundary of the phase, constrains the planning of some ‘free’ tasks to that phase. When you click the planning decision table on the boundary of the phase, the ‘free’ tasks, which planning is constrained to that phase, are highlighted (Tasks 2 and 4). Task 3 can be planned anytime while the case is active. Only the execution is constrained to the phase it is in (same as Tasks 2 and 4).

Task 2 can be planned ‘Any Time’ (free/ad-hoc) and can be planned as an ‘M’ follow-up planning decisions from Task 1.
### Requirement:

It should be possible to abstract hours accounting/labor costing information from the case system, based on activity instance monitoring data, including data such as norm time, percentage complete, etc. as well as presence/absence data of meeting activities, etc. Via process integration this data can be automatically processed into an ERP system.

### Appendix:

(6) Case Work Planning (Activity Overview)
(Norm duration, Norm Time, Planning Percentage)
(12) Case Tasks
(Task can Produce and Read/Update Case File Parts)

### ORM meta-model:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Norm Duration</th>
<th>Planning Percentage</th>
<th>Norm Time</th>
<th>Assignable Activity</th>
</tr>
</thead>
</table>

**Figure 60: (6) Case Work Planning (Snippet)**

<table>
<thead>
<tr>
<th>Task</th>
<th>Case Work</th>
<th>Case File Part</th>
<th>Produce</th>
<th>Read / Update</th>
<th>Tracked by / Tracks</th>
</tr>
</thead>
</table>

**Figure 61: (12) Case Tasks (Snippet)**

### ORC logic:

- 

### Notation:

- 

### Note:

This is part of the run-time framework. The data corresponds to run-time data (instances) and cannot be set in modeling environment. This should be taken to further studies.
Requirement: We can generalize the use of activity clusters. E.g. two types of clusters. One type as described above: Follow-up to an activity cluster with the following semantics: zero or one activity instances can be planned based on zero or one activities in the cluster. The other type is plan activities in the cluster. Follow-up to an activity cluster, with the following semantics: An activity instance is planned for any cluster. Typically this cluster would also contain one or more (project management type of) dependencies between activities in the cluster.

Appendix: (6) Case Work Planning (Activity Overview) (Fragment and Pallet) (10) Case Decision to Fragment Constraint (Decisions on boundary of Fragment)

ORM meta-model:

**Figure 62:** (6) Case Work Planning (Activity Overview) (Snippet)

**Figure 63:** (10) Case Decision to Fragment Constraint (Full)

ORC logic: -
**Notation:**

<table>
<thead>
<tr>
<th>Task 1</th>
<th></th>
<th>Task 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 4</th>
<th></th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Task 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Task 7</td>
</tr>
</tbody>
</table>

**Figure 64: Pallet and Fragment Notation**

**Note:** The pallet is depicted by a color pallet in the left upper corner. Here you can plan zero or one instances based on zero or one tasks in the ‘Pallet’. The Fragment is depicted by a scissors-icon in the upper left corner. This is more like a plan fragment. When you plan this ‘Fragment’ there will be another planning thread running next to the current planning thread of the case. The green dot in the ‘Fragment’ is the ‘Fragment Initiated Handler’ this is similar to a ‘State Entry Handler’. The dashed arrow in the upper right corner depicts that there are some dependencies between the tasks inside the ‘Fragment’. These dependencies can be set in the properties of the ‘Fragment’. There is also the possibility to place a ‘Planning Decision Table’ and/or ‘Enabling Decision Table’ on the boundary of the ‘Fragment’. The same as with phases.
5.2 Expressing a Scrum project with the suggested notation

Now that all the requirements have been mapped on the proposed semantics, it is time to see if a Scrum case can be designed using the proposed notation and following the proposed meta-models and logic. As stated before, this study is related to the PD automation project and the two parts: Analysis & Design and Implementation Project. As a reminder, the following picture illustrates the relationship between these two ‘cases’ (when we look at the product innovation ecology):

![Diagram](source: [SWE10])

The Analysis and Design case ‘creates’ and ‘releases’ the Implementation Project case. On the other hand, the Implementation Project case sends an ‘analysis feedback’ back to the Analysis and Design case.

Before showing the new Scrum case models, the following table shows a summary into the proposed notation which has been used in order to create the Scrum case models.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Initial Case Icon](source: [SWE10])</td>
<td>This icon is used for an Initial Phase or Start Case.</td>
</tr>
<tr>
<td>![Activity View Icon](source: [SWE10])</td>
<td>This icon is shown on an activity to show it is used to view a plan, like a Gantt or Burn Down chart. MS Project like view. But fully interactively.</td>
</tr>
<tr>
<td>![Activity Cluster Icon](source: [SWE10])</td>
<td>With the new notation (and suggested meta-model) it is now possible to have an activity cluster as a follow-up.</td>
</tr>
<tr>
<td>![Guard Icon](source: [SWE10])</td>
<td>This icon is used to show if there is a ‘guard’ on a transition. Rules on the transition, If … then Transition.</td>
</tr>
<tr>
<td>![Hand Icon](source: [SWE10])</td>
<td>The hand icon is used to show that an activity is ‘free’ to plan ‘any-time’. This was already possible, but now it is more visual.</td>
</tr>
</tbody>
</table>
5.2.1 Analysis and Design Case Model

The following picture shows the case model (based on the proposed semantics) of the Analysis and Design case.

![Analysis and Design Case Model](image)

Figure 66: Analysis and Design Case Model
The case model depicted above, shows how the Analysis and Design case ‘Creates’ and ‘Releases’ the Implementation Project. This is done during the ‘Estimation Meeting’. This meeting will be planned and executed during the ‘Analyze Feedback’ or can be planned and executed ‘any time’ (free).

The ‘Link And Estimate Story’ task will let the user provide points to stories. A story is a short formulation of a requirement (or need) set by the customer. Where a requirement might affect different teams, a story is a specific requirement for one team. During this task the team can assign ‘story points’ to a story. A story point is a measure to estimate effort or relative size of tasks or stories. These story points are assigned to a story during a ‘planning poker’. Planning poker is based on a list of user stories and a deck of numbered cards. Mostly the numbers: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40 and 100 and a ‘?’ (if someone doesn’t know how much points to assign). During some time the team will debate about a user story and every team members shows, with a numbered card, how many story points he/she would assign to the user story currently being debated. The more story points a user story receives, the more effort and the bigger the size of a user story.

There are two more ‘free’ activities which can be planned (one or more instances) during the entire ‘Analysis and Design’ case. During the ‘Update Product Backlog’ task the items on the product backlog can be mutated. Items can be added, modified or deleted, or the way the items are prioritized might change. This is an ongoing task, therefore this task has been made ‘free’. The other ‘free’ task is the ‘Create wave on Story’ task. This tasks allows to build consensus about a user story.

5.2.2 Implementation Project Case Model

The next figure depicts the case model of the Implementation Project case. Also here is the communication with the Analysis and Design case made perfectly clear. The Implementation Project is ‘Created’ (see transition to Waiting for Release phase), after that the Implementation Project can be ‘Released’. This is the communication from the Analysis and Design case to the Implementation Project case. The communication from the Implementation Project case to the Analysis and Design case, the ‘Analysis Feedback’, is shown by the ‘free’ activity ‘Analysis Feedback’. When we look back at the Analysis and Design Case Model, we see a local event handler which waits for a ‘New Analysis Feedback’. And here we have our closed loop between these two cases.
Figure 67: Implementation Project Case Model
Next to create a closed loop between the two cases ‘analysis and design’ and ‘implementation project’, there is also some traces to follow inside this case model.

After a ‘Implementation Project’ has been created and released, the case can start with its first ‘Sprint’ (the first ‘real’ state). The sprint starts with a ‘Phase Entry’ event handler. This event handler will automatically plan (the follow-up with an ‘A’ on it) a ‘Sprint Planning Meeting’. This task is a meeting activity and will automatically invite people and track absence. This meeting activity should be closed before the ‘Sprint’ phase is left (this is indicated by a small red marker in the lower right corner). During the ‘Sprint Planning Meeting’ several tasks can be planned and executed. This is visualized by the ‘I’ (intermediate) follow-up from the ‘Sprint Planning Meeting’ to other tasks. These ‘I’ follow-ups will not suspend the ‘Sprint Planning Meeting’. The option to suspend the tasks it came from, can be set in the options of the ‘I’ follow-up. Just by right clicking the follow-up and choose for properties.

The tasks which can be planned and executed (the order in which these tasks are depicted in the model does not say anything about the order in which these tasks should be planned and executed, this is free) during the ‘Sprint Planning Meeting’ are:

- ‘Enter capacity’
  - This task allows to set the capacity for the current Sprint. How many team members are available and for how many hours. It can happen that e.g. some team members work two days on this project and three days on a other project. Or that a team member has some day in the week off.

- ‘Track “Done” Criteria’
  - This task involves several other small tasks which will produce everything something else. These smaller tasks are: Formulate the Sprint goal (what should be delivered at the end of the Sprint) and compare realized objectives with set objectives. Where the latter might not occur during the ‘Sprint Planning Meeting’, but more during the ‘Daily Scrum Meeting’.

- ‘Update Sprint Backlog’
  - This tasks also includes several other small tasks, such as: ‘create the sprint backlog’, ‘prioritize the sprint backlog’, ‘divide work into 1-3 working days’ (decide on amount of work), ‘commit to work’ (let the team commit to the amount of work to be done in the sprint) and ‘explain user stories and ensure understanding’.

- ‘Prioritize Sprint Backlog’
  - This task can be planned (one or more instances) anytime during the ‘Sprint Planning Meeting’ and the ‘Daily Scrum Meeting’ to re-prioritize the sprint backlog. This might change from time to time because e.g. the customer has re-set the priorities of some user stories.

During the ‘Update Sprint Backlog’, the actual planning of design, code and test tasks are planned. After these tasks are planned, the capacity is set and the work has been divided into 1-3 working days (and maybe the process has been tracked e.g. in order of percentages), a burn down chart or a Gantt chart can easily be created. More about this further in this section.

The actual designing, coding and testing tasks, can change from project to project. The designer of the actual Scrum case models is free to make more concrete tasks here. The ‘I’ follow-ups from the ‘Update Sprint Backlog’ tasks are not connected to any design/code/test task in particular, but to the ‘pallet’ activity cluster.
surrounding the actual design/code/test tasks. So a user is free to plan the tasks which are necessary to complete the features to be delivered during that ‘Sprint’. The execution of these design/code/test tasks is not really constrained to the ‘Sprint’ phase (there is no marker in the lower right corner). So when a task is planned (and maybe currently is executed) will not be unplanned (or cancelled when in progress) when the phase ‘Sprint’ is left.

Every 24 hours, there will be automatically a ‘Daily Scrum Meeting’ planned. This is depicted by the timer event handler and a ‘A’ follow-up to the meeting task ‘Daily Scrum Meeting’. Also this meeting should be finished before leaving the ‘Sprint’ phase. During this meeting several tasks can be planned and executed. These tasks are:

- ‘Track “Done” Criteria’
  - See task description under ‘Sprint Planning Meeting’
- ‘Update Sprint Backlog’
  - See task description under ‘Sprint Planning Meeting’
- ‘Report Deviations’
  - During this task the team members tell what has been done the last 24 hours, what will be done the next 24 hours and what blocking issues they have encountered. This might involve several deviations from the original plan. During this task it is possible to set the progress of the planned (and maybe executed) tasks. Maybe a task might take longer or shorter than expected. This task might produce an updated plan.
- ‘Initiate Corrective Actions’
  - When some deviations or blocking issues have been reported and might endanger the features planned to be delivered in the current sprint, some corrective actions can be taken. This might involve taking less time for some other features to be delivered etc.

Next there are two ‘free’ activities in the ‘Sprint’ phase: ‘Log Issue’ and ‘Administer Sprint’. The ‘Log Issue’ tasks enables the ‘Issue’ case (see the product innovation ecology), but this is no part of this research. The ‘Administer Sprint’ allows for the Scrum master and/or team to administer the current sprint. What this tasks produces, can be set by the designer of the Scrum case model. Not only the execution, but also the planning of these ‘free’ activities is constrained to the ‘Sprint’ phase. This is shown by the planning decision table icon on the phase boundary. The properties of this planning table will show which activities are bound to this planning table and can only be planned while in that phase.

After the time of the sprint has elapsed (at Cordys this is two weeks, but this can be set on the properties of this transition), the sprint will be ended the case will be transferred to the ‘Sprint Review’ phase. The ‘Sprint Review’ phase will start with the automatic planning of the meeting tasks ‘Sprint Review Meeting’. During this meeting the following tasks can be planned and executed:

- ‘Evaluate “Done” Criteria’
  - This is about the same as the ‘Evaluate “Done” Criteria’ task in the ‘Sprint’ phase. During this task the features which are truly “done” designed, coded, tested and documentation has been written) are compared to the features which were planned during the ‘Sprint Planning Meeting’. This task is a mandatory task, this is depicted by the red exclamation mark in the task.
- ‘Add New Topics To Product Backlog’
  - When some features have not been realized during the past sprint, and should have been realized, they are transferred back to the product backlog. The product owner might also have some
additional needs, requirements or user stories, these items can also be added to the product backlog.

- ‘Sprint Retrospective’
  - This, mostly, is the last task to be done during the ‘Sprint Review Meeting’ and this does not include the product owner. Where the product owner is involved in the first two tasks. During this retrospective the past sprint is being evaluated and the enhancements for the next sprint are being reported.

Also this phase has a ‘free’ task, where the planning of this task is also constrained to this phase (see the planning table on the phase boundary). The ‘Log Issue’ task is the same as the ‘free’ task ‘Log Issue’ in the ‘Sprint’ phase.

When the set “done” criteria (i.e. all the items on the product backlog) have not been met, the case will return to the ‘Sprint’ phase and a next sprint is being planned. When the “done” criteria have been met, the case can be closed.

There are also some ‘free’ tasks which can be planned during the entire life cycle of the case. These activities are not constrained by any phase, but rather to the ‘root phase’ (the white background) of the case life cycle, these tasks are:

- ‘View Gantt Chart’
  - This task should be linked to some application like MS Project, this application will show the planned tasks and how far they are ‘done’ (e.g. in percentages) and will illustrate this in a Gantt chart like view.

- ‘Analysis Feedback’
  - This task allows the user to make an analysis feedback and send it to the ‘Analysis and Design’ case.

- ‘Specify Story’
  - When a story lacks some specific details, this task allows to specify a story more so that the team will commit to it.

- ‘Specify Need’
  - The same as the above task, but related to a need from the customer.

Next to these ‘free’ tasks, there is also a task which executes a predefined workflow: ‘Process Hours Accounting’. This predefined workflow can manually be planned and then executed according to a schedule or e.g. every hour or every two weeks.
6. Next steps in Case Management

The scope of this study was the modeling environment of Case Management. Therefore there are still some uncovered areas. The first section of this chapter (6.1 Next steps for completing Scrum support) will suggest which further studies are necessary to achieve a full support of Scrum in Case Management. The last section of this chapter (6.2 Next steps for completing standard of Case Management) will show how the suggestions provided in this thesis can be improved for which there was no time during this study.

6.1 Next steps for completing Scrum support

Because the scope this study was the modeling environment of Case Management, some requirements couldn’t be fully mapped (see 5.1 Mapping of the requirements on the proposed semantics). This was because some requirements are based on the run-time of Case Management. So when a case is in progress. These requirements were:

- More dynamics is required on case task UI in run-time. Instead of repeatedly having to switch between task form and inbox, it is better to e.g. automatically generate an hyperlink or activate some form control on the task UI, at the moment follow-up is planned from that activity and can be executed directly while being in that activity. Based on which the user can start the planned activity directly in context of the overall activity. This is e.g. extremely important during various Scrum meetings
- Next to specific task, also support task templates. When follow-up is planned interactively, specific tasks are planned based on these templates. By the way, follow-up relationships can point to task templates. Task templates might define: task type/spec and link to a tool. Task templates might further define defaults for norm times, durations, etc. (this is supported by the meta-model)
- Use an interactive Gantt chart as view on the run-time case process and its activities in the Case Instance Monitor. The Gantt chart view can also be used during interactive planning, to insert new activities (and probably dependencies). If people are authorized they can also update activities, via the Gantt chart, at any time, e.g. to update/delete activities. Updating activities includes setting percentage complete, etc. (this is supported by the meta-model)
  - And also belonging to this requirements:
    - Gantt chart can also be viewed via an activity (e.g. view Gantt chart activity)
- Functional modeling of case file with case file parts, aligned with ‘entity modeling’
- It should be possible to abstract hours accounting/labor costing information from the case system, based on activity instance monitoring data, including data such as norm time, percentage complete, etc. as well as presence /absence data of meeting activities, etc. Via process integration this data can be automatically processed into an ERP system

The next sections will give some examples of how these requirements can be met in run-time.
6.1.1 Dynamics in case task UI

As the example of run-time of the current Case Management solution of Cordys showed (see 3.2.4 Example in Run-Time (current implementation of Cordys Case Management)), when a task is planned it pops up in the inbox and can be started from there. But this takes too much time when someone plans (and most likely directly wants to execute) a task during a meeting. A simple solution can be a small hyperlink (or button) from the meeting user interface (UI) from which a user can directly plan and execute tasks during the current running meeting.

6.1.2 Task templates and open with

To support task templates, it is possible to re-use a template for different tasks. This makes it easier for a design user to get the same view for all the kind of tasks in a case life cycle. These templates can be made by some sort of task template modeler. When a template has been made, it can be applied to all or some tasks.

Next to these templates, it should also be possible to link a task to another application. This makes it possible to link to e.g. a Word or excel document which should be produced with that specific task. This word or excel document is a template which can be filled-out by the run-time user. It should not only be possible to produce certain documents, but also be possible to consume a document. E.g. link to a pdf document or a power point presentation for more information about a certain task. Or when more data is required during a task (the intermediate follow-up with suspension) there can be linked to a document with more information (e.g. insurance policy).

6.1.3 Interactive Gantt chart

The Gantt chart is a standard view to view a schedule of e.g. a case or project. The following picture is an example of a standard Gantt chart view (where WBS stands for Work Breakdown Structure, but has no direct meaning for this thesis, but just is used as an example of a Gantt chart like view).

![Gantt chart example](http://en.wikipedia.org/wiki/Gantt_chart)
There should be a possibility (e.g. by the ‘Edit Work Plan’ activity proposed in this study) to produce a Gantt chart view of the course of the case. This can be done by linking the ‘Edit Work Plan’ activity to a MS Project like application (maybe MS Project itself) which can generate such a Gantt chart view. Next to a view of the current case course (which can be used to produce a ‘Burn Down Chart’; for more information about burn down charts see 3.1.2 How Scrum works in practice) it should be possible (for people with the right rights to the system) to mutate tasks in this Gantt chart. This might involve:

- Add tasks to the planning;
- Delete tasks from the planning;
- Cancel tasks that are in progress;
- Modify e.g. the % complete of a task.

6.1.4 Functional modeling of Case File

Next to the modeling environment modeler (and the task template modeler introduced previously in 6.1.2), there should also be a Case File modeler. The modeling environment user should be able to model the case file belonging to that case and the corresponding case file elements and attributes. This modeler can be based upon Figure 44: Case File Modeling Notation. When a Case File is correctly modeled, it should be possible to link certain tasks to the case file parts in the case file, which should/can be produced or consumed by which tasks.

6.1.5 Abstracting data from Case File

When a correct case file has been modeled, it should be easy to extract data from the case file (e.g. hours accounting, labor costs, percentage complete etc.). This data can then be used for the Gantt chart view, but also be used by some kind of ERP application. This way it is possible to create several (management) reports and get a better view of the progress of the case.

6.2 Next steps for completing standard of Case Management

As was stated before, the semantics which are proposed in this study were also discussed with several OMG co-submitters. The first discussion some good comments on this proposal. These comments can be found in this section. To reach total consensus about the proposed semantics with OMG will take a good deal of time (because there are many other companies involved) and therefore will not be part of this study. This section will provide an overview of what enhancements were suggested.

Because it can be sometimes a burden to 100% fix things in the modeling environment, some case-specific planning should be made possible in run-time. This research did not focus on the run-time of Case Management, another study could be very helpful to refine also the run-time of Case Management (for the new standard but also for the current Cordys implementation).

In run-time and in a Gantt chart view, are constrainable moments important. Such constraints between two (or more) activities can be: Finish-Finish, Start-Finish, Finish-Start and Start-Start. But also: Start on, Finish on, etc. It is not possible to set these constraints in modeling environment, because modeling environment is about abstracts of activities. These constraints can only be set between instances of these activities. E.g. a ‘free’ activity can be planned multiple times, so between which instances are these constraints set? The suggested meta-model can still be
applied if these constraints (which can be set in modeling environment) are used as guidance. A default setting, when a constraint is been set between activities in run-time.

Another aspect that is missing from these semantics is security. The most important aspects of security, described by Tanenbaum (2002, [TAN02]), are: Confidentiality (‘information only disclosed to authorized parties’), Integrity (‘modifications only possible by authorized entities’) and Availability (‘resource accessible whenever an authorized party needs access to it’), or simply said CIA. These CIA features should be taken into consideration when security will be introduced into the standard of Case Management. Another aspect of security, also described by Tanenbaum [2002, [TAN02]], is non-repudiation. Or simply said: ‘Prevent entity from denying previous commitments or actions’. This security feature has been introduced in the proposed semantics. The following figure shows a snippet of the ‘Case Work Planning (Activity Overview)’ meta-model.

![Figure 69: Case Work Planning (Activity Overview) Snippet](image)

This meta-model does not fully support the ‘non-repudiation’ feature, but an activity can be tracked by a unique electronic signature of a case worker.

There was also another comment on the proposed semantics, re-usability. It should be possible to re-use activities, snippets from other cases or re-use whole cases. Now this can only be done by some form of task template (see in previous section: 6.1.2 Task templates and open with). This may be realized by storing entire tasks (in some form of global template) or by using some form of categories (for activities and cases itself). So when a modeling environment user designs a new case model and adds an activity to the grid, he/she can choose between several activity categories. The same counts for the case file beneath the case. This is also important because some activities produce or consume case file parts, so the case file should fit the activity category.
7. Conclusion

The former goal of this research was to refine the modeling environment of the current implementation of Case Management at Cordys to fully support Scrum. This goal was stretched a bit further when a request for proposal (initiated by OMG) reached Cordys. The new goal was to propose semantics for a standard of Case Management and also to propose these semantics to OMG as a standard to be used for Case Management.

These proposed semantics were validated with the Scrum requirements and it was shown that these proposed semantics could fully support Scrum. With the exception of some requirements which are less dependent on the modeling environment of Case Management.

It can be concluded that the semantics, which are proposed in this thesis, are just a first step to a standard of Case Management. Chapter six (next steps in Case Management) shows which further studies should be performed in order to come to a full standard of Case Management.

The semantics proposed in this study are based on a standard for Case Management. Cordys could use these proposed semantics to implement these solutions to fully support Scrum via incremental extensions of the current Case Management solution at Cordys.

The semantics proposed in this research were also proposed to OMG. Because OMG has several (developing) members (next to Cordys), it would take a lot of time to reach total consensus about the proposed semantics. So this is no part of this study, this is still an ongoing process. The proposed semantics in this research are a first step for Cordys to improve their current Case Management solution by selective porting them on it.
8. References


[SMA08] Smart PA et al., UNDERSTANDING BUSINESS PROCESS MANAGEMENT: Implications for theory and practice, Discussion Papers in Management, University of Exeter, 2008

[SUT04] Sutherland J, AGILE DEVELOPMENT:LESSONS LEARNED FROM THE FIRST SCRUM, ScrumAlliance, 2004

[SUT05] Sutherland J, Future of Scrum: Parallel Pipelining of Sprint in Complex Projects, Agile Conference, 2005


Appendix A: Current meta-model based on UML state machines
Appendix B: Meta-model current implementation Case Management of Cordys

Note:
This ORM meta model represents the design-time of the current implementation of Case Management in Cordys.

Subject defining rules:
- Final State Is State Identified by Vertex.
- Initial State Is Event Identified by Vertex.
- Initial State Is Vertex Identified by Event.
- Transition Is Event Identified by Vertex.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by State.
- Transition Is Event Identified by State.
- Transition Is Event Identified by State.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
- Transition Is Event Identified by Event.
Appendix C: Proposed ORM meta-models

(1) Case Root Model
(2) Case File
(3) Case Life Cycle
(4) Event
(5) Event Composition
(6) Case Work Planning (Activity Overview)
(7) Case Decisions
(8) Case Objectives

Case Objective — Measure

Constraint Rule

Constraint

(9) Case Life Cycle and Case Work Planning Combined.

Life Cycle Phase — Has / Belongs to Activity Execution

To Phase Constraint

UnPlanOnPhaseExit

Activity Planning Decision

Event Handler — Triggers / Triggered by

Phase Exit Impact Kind

{Proceed, Suspend, Terminate, None}
(10) Case Decision to Fragment Constraints

(11) Activity Planning Decision to Case Task Constraints
(12) Case Tasks

(13) Case Work Product
(14) Case Work Planning - Constrainable Sequence

Constrainable Sequence

Activity

Fragment

Case Work Planning

Activity Cluster

Constraint Kind

Lag

Constraint Kind Predetermined

MustBeConstrained

Constraint Kind

Moment

Constraint Kind Predetermined

MustBeConstrained

{ AsSoonAsPossible, ASAP, AsLateAsPossible, ALAP, NotEarlierThan, NLT, NotLaterThan, NLT, Exactly On }
(15) Case Organization

OSM = Organization Structure Modeling (under development by OMG)
VDM = Value Delivery Modeling (under development by OMG)
Appendix D: Subtype defining rules for proposed meta-models

### (2) Case File
- Case File Document IS Case File Element Identified by Case File Element Sort ‘CFD’
- Case File Association IS Case File Element Identified by Case File Element Sort ‘CFAs’
- Case File Part IS Case File Element Identified by Case File Element Sort ‘CFP’
- Case Attribute IS Case File Element Identified by Case File Element Sort ‘CA’
- Case File Annotation IS Case File Element Identified by Case File Element Sort ‘CFAn’
- Case File Part Association IS Case File Element Identified by Case File Element Sort ‘CFPA’

### (3) Case Life Cycle
- Local Event Handler IS Event Handler Identified by Event Handler Sort ‘LEH’
- Phase Entry Handler IS Event Handler Identified by Event Handler Sort ‘PEnH’
- Phase Exit Handler IS Event Handler Identified by Event Handler Sort ‘PExH’
- Fragment Instantiated Handler IS Event Handler Identified by Event Handler Sort ‘FIH’
- Case Created Handler IS Event Handler Identified by Event Handler Sort ‘CCH’
- Life Cycle Transition IS Event Handler Identified by Event Handler Sort ‘T’
- Pseudo IS Vertex Identified by Vertex Sort ‘Pseudo’
- Life Cycle Phase IS Vertex Identified by Vertex Sort ‘Phase’

### (4) Event
- Case File Event IS Event Identified by Event Sort ‘CFE’
- Disposition Event IS Event Identified by Event Sort ‘DE’
- Activity Event IS Event Identified by Event Sort ‘AE’
- Phase Event IS Event Identified by Event Sort ‘PE’
- Time Event IS Event Identified by Event Sort ‘TE’
- External Event IS Event Identified by Event Sort ‘EE’
- Composite Event IS Event Identified by Event Sort ‘CE’
- Relative Time Event IS Time Event Identified by Time Event Sort ‘Relative’
- Absolute Time Event IS Time Event Identified by Time Event Sort ‘Absolute’

### (6) Case Work Planning (Activity Overview)
- Assignable Activity IS Activity Identified by Activity Sort ‘A’
- Sub Process IS Activity Identified by Activity Sort ‘SP’
- Sub Case IS Activity Identified by Activity Sort ‘SC’
- Any Activity IS Activity Identified by Activity Sort ‘AA’
- Activity Cluster IS Activity Identified by Activity Sort ‘AC’

- Task IS Assignable Activity Identified by Assignable Activity Sort ‘T’
- Close Case IS Assignable Activity Identified by Assignable Activity Sort ‘C’
- Edit Work Plan IS Assignable Activity Identified by Assignable Activity Sort ‘E’

- Fragment IS Activity Cluster Identified by Activity Cluster Sort ‘F’
- Pallet IS Activity Cluster Identified by Activity Cluster Sort ‘P’

- Meeting IS Task Identified by Task Sort ‘M’
- Decision IS Task Identified by Task Sort ‘D’

(7) Case Decisions

- Activity Applicability Decision Alternative IS Decision Alternative Identified by Decision Alternative Sort ‘Ap’
- Activity Availability Decision Alternative IS Decision Alternative Identified by Decision Alternative Sort ‘Av’
- Activity Planning Decision Alternative IS Decision Alternative Identified by Decision Alternative Sort ‘P’
- Work Assignment Decision Alternative IS Decision Alternative Identified by Decision Alternative Sort ‘W’
- Disposition Decision Alternative IS Decision Alternative Identified by Decision Alternative Sort ‘D’

- Activity Decision IS Decision Identified by Decision Sort ‘A’
- Work Assignment Decision IS Decision Identified by Decision Sort ‘W’
- Disposition Decision IS Decision Identified by Decision Sort ‘D’

- Activity Enabling Decision IS Activity Decision Identified by Activity Decision Sort ‘E’
- Activity Planning Decision IS Activity Decision Identified by Activity Decision Sort ‘P’

- Activity Applicability Decision IS Activity Enabling Decision Identified by Activity Enabling Decision Sort ‘Ap’
- Activity Availability Decision IS Activity Enabling Decision Identified by Activity Enabling Decision Sort ‘Av’