MASTER THESIS

MANAGING REQUIREMENTS IN BUSINESS PROCESS MANAGEMENT SUITE PROJECTS

Public version

Chris Aukema

15-07-2011

UNIVERSITY OF TWENTE.

Master Thesis Chris Aukema

A Requirements Management Approach for Business Process Management Suite Projects

Utrecht, July 2011

Author

Chris Aukema

Program	Master Business Information Technology, School of Management and Governance
Student number	0110531
E-mail	c.h.aukema@alumnus.utwente.nl

Graduation Committee

Maria Iacob

Department	University of Twente, School of Management and Governance
E-mail	m.e.iacob@utwente.nl

Marten van Sinderen

Department	University of Twente, Computer Science
E-mail	m.j.vansinderen@utwente.nl

Supervisor Capgemini

Willem Schellekens

Department	Capgemini FS GBU
E-mail	willem.schellekens@capgemini.com

PO box 217
7500 AE Enschede

PO box 2575
3500 GN Utrecht

Preface

Six months ago I started my master thesis of the University of Twente in Business Information Technology at Capgemini. This gave me the possibility to explore the twilight zone between the academic world and the business world.

As me and my fellow classmates repeatedly recalled over the past few years, this is where we like to be, in between. During my master thesis I was in between the academic and the business world, in between the business and the IT world and in between the Requirements and Business Process Management world. The friction that arises when combining two fields, I can only describe as exiting.

BPM asks for agility, speed and a "it's ok if it doesn't work, will fixed it soon enough" attitude. Requirements, by nature, ask for stability, agreement and a 'we would like to know everything in advance" attitude. Fortunately, reality is less black and white. I've enjoyed seeking new methods to tackle challenges and finding middle ground to create an approach suitable for practitioners struggling in this combined field.

Needless to say, I could not have done this research on my own. Luckily there were several people that supported and challenged me along the way, for which I would like to thank them. Starting with Maria and Marten, my university supervisors for their valuable advice, their quick action when needed and their trust. Then, Willem, my Capgemini supervisor for being a great sparring partner and challenging me to always take it one step further. Nienke en Leo, for assisting me and guiding me through the process. Special thanks to Ruud and the whole team of Cornelly, who although operating in a different department, involved me in their actions and were always up for discussion. I would also like to thank all reviewers and interviewees that contributed to this research.

Last but not least, I would like to thank Anouk, for her endless support, Ronald for giving me a place to stay the last couple of months and all family and friends who made my time in Enschede a time I'll never forget.

Summary

Incentive
Capgemini FS GBU is getting more often involved in the implementation of software solutions using Business Process Management Suites (BPMSs). The requirements management practice is one that is mastered in the Custom Software Development (CSD) field, but lacks of a common approach in BPMS projects. To improve this situation, the goal of this research is to create a tool independent requirements management approach for BPMS projects.

Recommendations
The goal is reached, a useful, ease to use and complete requirements management approach for BPMS projects has been developed and is recommended to be used. It is a combination and adaption of existing methods, tailored to BPMS project use. It is organized into three main elements:

1. Be Agile, which focuses on creating an agile mindset, an agile project process and agile team members.
2. Collaborate, which focuses on the importance and possibilities of collaboration in a BPMS project. Especially regarding the topics: customer collaboration, collaborative requirements elicitation, collaboration of systems and offshoring.
3. Deliver, which focuses on the products of requirements management. It offers building blocks to help select the right deliverables for a project. It furthermore advices on the related topics: Vision document, connection of use cases to process models, traceability, prioritization, changing requirements, a lean requirements management plan, frameworks and templates.

Motivation
The approach is based on theory as well as practice. The state of the art has been explored, researching traditional approaches, agile approaches and BPMS vendor approaches. They all offered advantages as well as disadvantages for use in BPMS projects. To explore practice, eighteen interviews were conducted, to learn what methods from theory are actually used, what best practices are applied and what problems still exist.

The combination of theory and practice led to a requirements management approach combined and adjusted for use in BPMS projects. This approach was reviewed and in general it was found useful, easy to use and complete. The recommendations the reviewers made led to an improved version of this approach.

Consequences
Application of the requirements management approach for BPMS projects will lead to a better aligned requirements management process, as opposed to using existing methods. The requirements process as well as its products will be in gear with the needs and possibilities of working with a BPMS.

Table of contents

Table of figures

Table of abbreviations

BI	Business Intelligence
BPA	Business Process Analyzer
BPM	Business Process Management
BPMN	Business Process Management Notation
BPMS	Business Process Management Suite
CSD	Custom Software development
DCO	Direct Capture of Objectives
ETL	Extraction, Transformation and Load
FS GBU	Financial Services Global Business Unit
IRMA	Integrated Requirements Management Approach
IT	Information Technology
OpenUP	Open Unified Process
OTOPOP	One Time, One Place, One Person
RFC	Request for Change
RM	Requirements Management
RUP	Rational Unified Process
SEMBA	Structured Expert Method for Business Analysis
SOA	Service Oriented Architecture
UCD	Use Case Description
UML	Unified Modeling Language

1 Introduction

1.1 Background

This section describes the background of the perceived problem by first describing the current state of requirements management (RM) and its importance and second describing the current state of business process management (BPM) and Business Process Management Suites (BPMSs).

1.1.1 Requirements management

Definition

Everyone who has ever been in a software project knows this: Requirements start out simple, but can turn into a nightmare! Even if you, as a business analyst or system engineer, developed the skills to ask the right questions to your client, the right way, at the right time, you'll never know if the answers you'll get are the correct ones. When the same question is asked at a later moment in time, the answer will probably be different. A paradigm to deal with these problems is requirements management. (Leffingwell & Widrig, 2000) define a requirement as a capability that the system must deliver, either because it is needed or wanted by the user or because it is imposed by formal documentation, such as contracts or standards. (Leffingwell & Widrig, 2000) define requirements management as: *"A systematic approach for eliciting, organizing and documenting the requirements of the system, and a process that establishes and maintains agreement between customer and the project team on the changing requirements of the system."*

Approaches

Different authors ((Leffingwell & Widrig, 2000); (Nuseibeh & Easterbrook, 2000); (Kruchten, 2003)) use different approaches to requirements management, but generally they all acknowledge the same challenges/activities in the process. Roughly these include the following:

- Analyzing requirements; this is also called requirements engineering. Requirements need to be elicited from stakeholders; problems and wishes should be made clear.
- Organizing requirements; in practice it has shown to be impossible to fullfill all requirements on time and on budget, organizing requirements is about prioritizing and scoping the requirements in the project.
- Documenting requirements; when requirements start to increase in numbers it is important to document them in a structured way to be able to maintain them, trace them, refine them if needed and see who is responsible.
- Communicating requirements; make sure requirements are communicated between developers and stakeholders, so there is agreement about the requirements and they can be validated.
- Handling changing requirements; requirements will always change during a project and this should be handled correctly. Traceability and

communication can help doing this, by for instance showing the impact of a change.

As an extension, the requirements workflow of the Rational Unified Process (RUP) description by (Kruchten, 2003) also makes use of roles and artifacts (e.g. vision documents) to accompany the requirements management activities.

1.1.2 Business process management

Definition

Business process management is becoming more important in system development as it integrates business processes, information and information systems. It has interested communities in both the business administration field as well as the computer science field and is therefore located at the crossroads of business and IT. According to the book of (Weske, 2007) a business process is defined as: *"a set of activities that are performed in coordination in an organizational and technical environment. The activities jointly realize a business goal"*. Then (Weske, 2007) defines business process management (BPM) as: *"the concepts, methods and techniques to support the design, administration, configuration, enactment and analysis of business processes."* An important note made by (Aalst, Hofstede, & Weske, 2003) is that, by this definition, BPM is limited to operational processes. BPM needs to have information about the operational processes at hand. Therefore strategic level processes are excluded.

BPM Suites

A business process management system is defined by (Aalst, Hofstede, & Weske, 2003) as "*a generic software system that is driven by explicit process designs to enact and manage operational business processes*." The process designs are often graphically represented and the focus is on structured processes that need to handle a great number of cases. According to (McCoy & Cantara, 2010) a BPM suite (BPMS) supports the entire process improvement life cycle. Ranging from process discovery, definition and design to implementation, monitoring and analysis, and through ongoing optimization.

There are several of these BPM suites available on the market like Pega BPM (Pegasystems Inc., 2011), IBM Websphere process server (IBM Corporation, 2011), Mendix Business modeler (Mendix, 2011), BeInformed (BeInformed, 2011) etc. All these suites tend to do the same thing in essence, namely supporting business process management, but differ in their approach.

BPM development

In his book (Weske, 2007) states that BPM uses a short business process lifecycle, which has four phases: Design & Analysis, Configuration, Enactment and Evaluation. See Figure 1. It is stated that you enter the lifecycle at the design & analysis phase, which suggests there is no sole requirements phase, but the design of the new process is started right away. A white paper implementation guide of IBM BPM (Bergland, Maquil, Nguyen, & Son, 2009) does not indicate otherwise. The design & analyses phase is mostly based on business goals and storyboards. Developers tend to capture roles and identify process steps as candidates for business rules. This process relies on continuous refinement and on the experience of the developer.

An explanation for this way of developing may be that BPM driven development is a relatively new field. It is at this moment mostly used in smaller software projects, but the usage is expanding. Another explanation may be sought in the nature of BPM. It carries the values of an agile development project (Beck, et al., 2001): Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan.

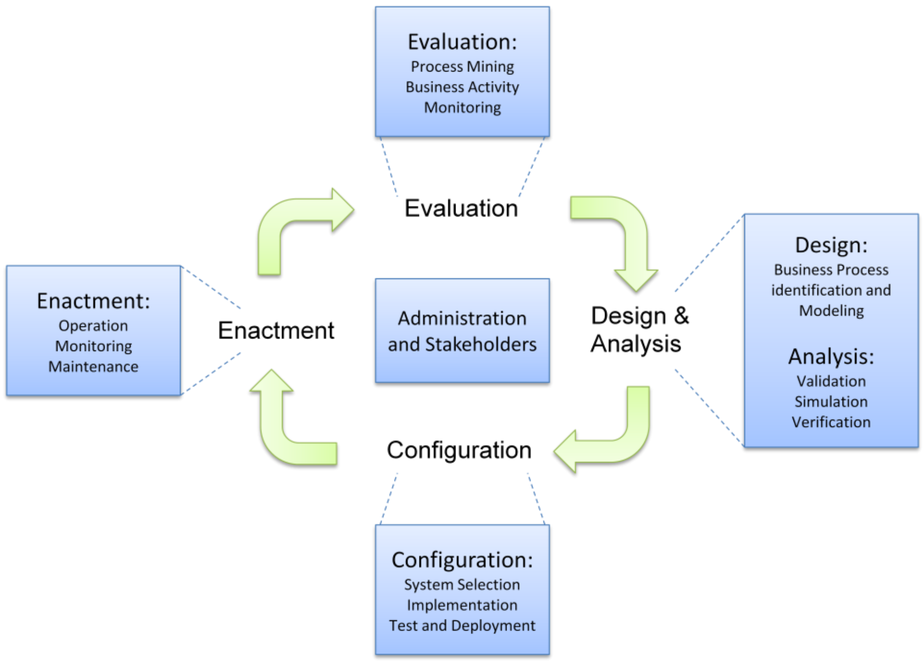

Figure 1: Business Process Lifecycle (Weske, 2007)

1.2 Problem statement

Although requirements management is as old as traditional software development itself, it is not fully developed in software development using business process management suites. In practice it is either the case that requirements are elicited traditionally (using for instance signed off use cases) and only when this phase is formally closed the BPMS design will start (1), this phenomenon can be seen at large traditional oriented customers such as banks. In other projects it can be seen that a very agile form of requirements management is used (2) which then has a high correlation with the design & analysis phase. See also Figure 2.

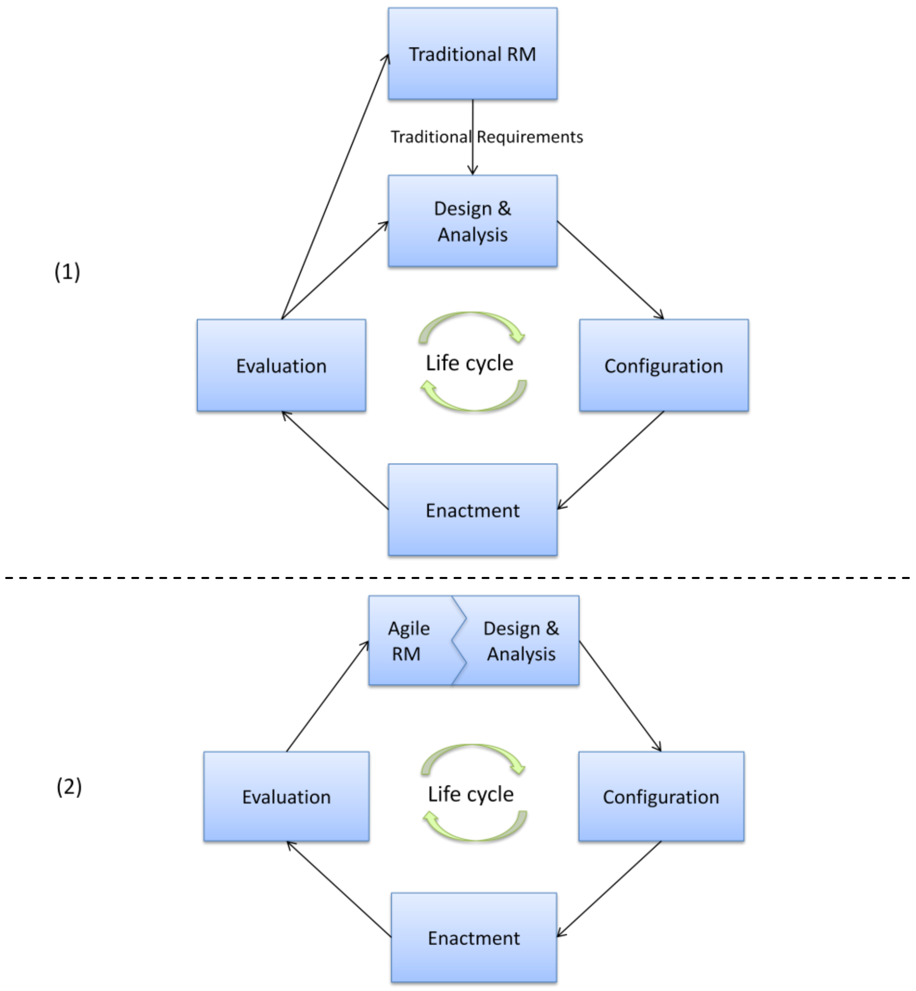

Figure 2: Possibilities As Is

The first option causes problems, because of mismatches between agile and waterfall development. The second is the way it is meant by a lot of BPMS suppliers. This approach is very tool dependant and not always applicable. Because BPMS projects have their own values and ways of development, the ideas of requirements management from neither traditional development nor agile development can be mapped one to one on most of the BPMS projects.

In a other words, the problem is that there is no single approach that gives methods, techniques and overall guidance for managing requirements in BPMS projects independent of tools or the project's nature.

1.3 Objectives & scope

The objective of this study is to close the gap described at the problem statement, namely: there is no single approach that gives methods, techniques and overall guidance for managing requirements in BPMS projects independent of tools or the project's nature. At Capgemini they recognize this problem not only for requirements management, but for the whole BPMS implementation cycle. After developing IRMA for traditional requirements management and SEMBA (Structured Expert Method for Business Analysis) for business analysis, they have now started a project to develop a tool independent best practice approach for implementing BPMSs. This project does not only cover researching requirements management for BPMS projects, but also all other phases of the development cycle. This research will contribute to this project by looking at the tool independent best practices for managing requirements in BPMS projects. Eventually creating a BPMS method containing a specific BPMS requirements management approach (see also Figure 3).

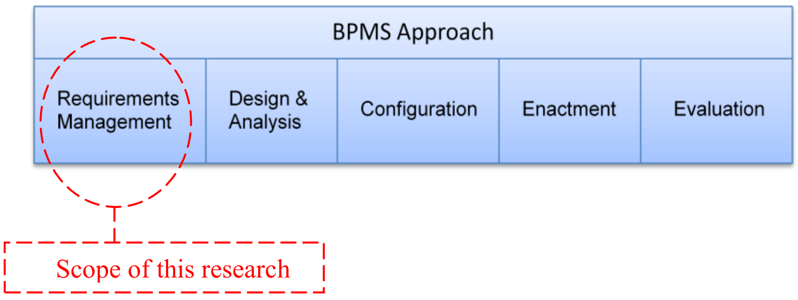

Figure 3: Scope

The scope of this project is to look at the requirements management phase in BPMS project approaches. This will be done by looking on one hand at existing requirement management techniques, traditional techniques as well as agile techniques, and on the other hand at approaches currently used in BPMS projects. When researching traditional requirements management the focus will be on IRMA, because this is an already validated approach constructed on standards as RUP and ISO 9126 (Klabbers, Spier, Zijlstra, & Aalberts, 2011). When researching current best practices in BPMS projects, the scope will be on three widely used BPMSs within Capgemini. These are: Pega systems, IBM websphere process server/Lombardi and Oracle BPM. All three are large worldwide used systems.

1.4 Relevance

Although the case is performed at Capgemini, for which the relevance is described above, the scientific relevance of this study is much broader. BPMS projects are performed more often, as companies start to see the benefits of this discipline. A BPMS is able to reduce costs, increase productivity and provide agility (McCoy & Cantara, 2010). The success of a software project, in our case a BPMS software project, is highly dependent on requirements. Three of the four most important reasons for software project failure are: Lack of user input, incomplete requirements and changing requirements (Schwalbe, 2007). Even though a BPMS is designed to cope with changes, these problems still exists. The aim of this study is to illustrate how to adapt requirements management to be more successful in BPMS projects, but also to explain how to use a BPMS to improve speed and quality of requirements management. By reducing the problems and improving the requirements management process this research impacts theory as well as practice.

1.5 Approach

The previous sections stated the why of this research, this section describes the what, where and how. The first part describes the company where the study is performed, the where. The second part describes the research questions that need to be answered to reach the goal, the what. The third part describes the research methodology to answer these questions, the how.

1.5.1 The company

Capgemini (Capgemini, 2011a) is one of the largest consultancy, technology, outsourcing and local professional services companies in the world. It was founded in 1967 in Grenoble (France) by Serge Kampf. With over a 106 000 employees working all over the world and 6000 employees working in the Netherlands, it performs IT services of all sizes and on all locations. Capgemini has a lot of experience in BPMS projects as well as requirements management, which makes them very suitable for this study. Capgemini consists of four major divisions: Consulting, Technology Services, Outsourcing and Financial Services. This study will be performed in the 'Financial Services Global Business Unit', in the practice 'Technology Development and Integration', in the cluster 'IT Governance Improvement' and in the expert group 'Requirements Management'. As said the research will also cooperate with the BPMS project, which is performed by a different division.

1.5.2 Research questions

The goal of this research is to develop a tool independent best practice approach for requirements management in BPMS projects. To reach this goal the following main question has been specified:

"How can requirement management in BPMS projects be improved using a tool independent BPMS requirements management approach?"

This main question is divided into sub questions, the answers to these sub questions will add up to answer the main question:

1. *What requirements management methods and techniques currently exist?*
   a. *In traditional software development?*
   b. *In agile software development?*
   c. *What are the prescribed methods by BPM Suite vendors?*
   d. *Which of these methods and techniques are useful for BPMS projects?*
2. *How are requirements currently managed in BPMS projects?*
   a. *What are the approaches used in practice?*
   b. *What are the problems encountered?*
3. *What is the recommended tool independent approach for requirements management in BPMS projects?*
   a. *What requirements management principles can be applied?*
   b. *How can they be applied?*
4. *What is the validity of the method?*

1.5.3 Research methodology

Using the classification theories of (Gregor, 2006), this research is classified as a design and action theory. *"The theory gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact."* In other words, it tells you how to do something. It gives a solution to a problem. A problem stated as several questions in the previous section, with a practical relevance as well as a theoretical relevance. To gather the data in order to answer the sub questions and eventually the main question, a combination of literature research and expert interviews is used.

To answer the first sub-question a literature study will be performed on traditional requirement management, managing requirements in agile development projects and requirements management as prescribed by BPMS vendors in white papers. The topic of traditional requirements management has existed for decades and it is expected that there is sufficient literature available. The topic of agile requirements management exists for a shorter period of time, but it is expected that there is lots of information available from theorists as well as practitioners. This will lead to the state of the art on requirements management. To answer the second sub question expert interviews will be conducted. Expert interviews will give inside in the best practices in BPMS projects as well as the problems and needs that practitioners have.

Using this information from the data gathering phase a tool independent Requirements Management Approach for BPMS projects will be developed. To test the validity of the method , it will be reviewed by experts and practitioners. A graphical representation of this research methodology can be found in Figure 4.

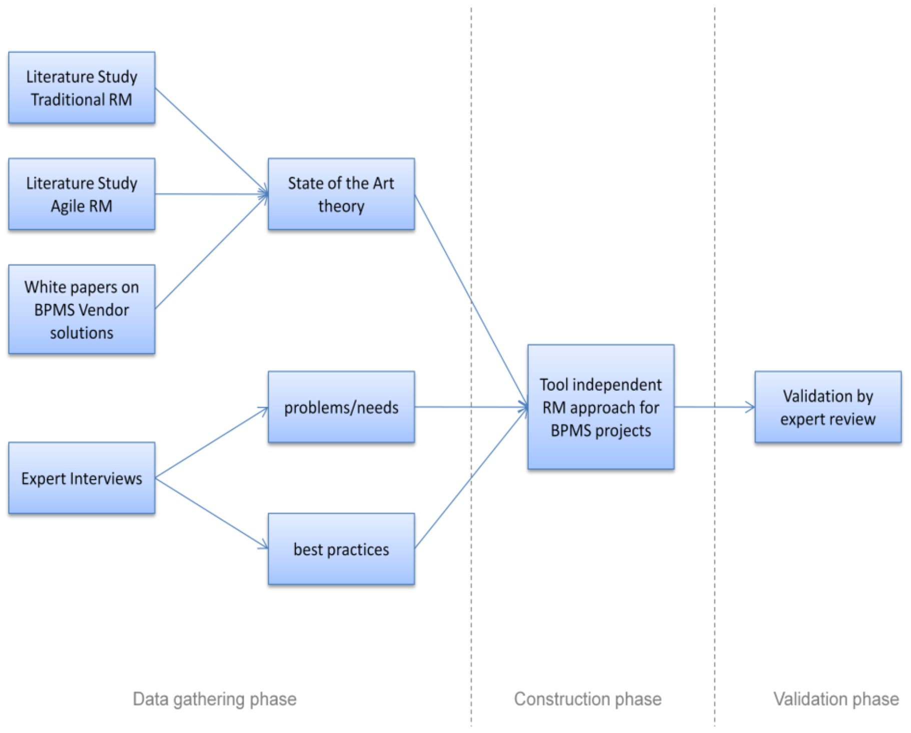
Literature Study Traditional RM
Literature Study Agile RM
White papers on BPMS Vendor solutions
State of the Art theory
Expert Interviews
problems/needs
best practices
Tool independent RM approach for BPMS projects
Validation by expert review
Data gathering phase
Construction phase
Validation phase

Figure 4: Research Methodology

2 State of the art

The goal of this chapter is to answer research questions 1:

1. *What requirements management methods and techniques currently exist?*
    a. *In traditional software development?*
    b. *In agile software development?*
    c. *What are the prescribed methods by BPM Suite vendors?*
    d. *Which of these methods and techniques are useful for BPMS projects?*

To answer these, this chapter describes the current state of the art approaches and methods that can be used to manage requirements. This is split up in three categories. First, traditional development approaches. Here the Rational Unified Process is reviewed and the Integrated Requirements Management Approach of Capgemini. Second, agile development approaches. Here the methods of Agile RUP, Open UP, Scrum and two visual requirements modeling techniques will be reviewed. Third, the BPMS vendor development approaches. Here the vendor specific approaches from Pegasystems, IBM and Oracle are reviewed. Per method the advantages and disadvantages are summarized. In the end the total literature research summary is given in Table 11 and conclusions are drawn.

2.1 Traditional approaches

2.1.1 RUP

The Rational Unified Process (RUP) is developed by the Rational Software Corporation, a division of IBM. The Rational Unified Process (Rational Software, 2003) is a software engineering process, delivered through a web-enabled, searchable knowledge base. RUP provides the team members of a development team with guidelines, tools and best practices. These best practices cannot be precisely quantified, but are generally used by successful organizations in the industry. The six most important best practices are:

1. Develop software iteratively
2. Manage requirements
3. Use component-based architectures
4. Visually model software
5. Verify software quality
6. Control changes to software

The best practice that this research focuses on is the second: Managing requirements. RUP describes how to elicit, organize, and document requirements. How to track and document tradeoffs and decisions and how to capture and communicate business requirements.

Overview

There are two dimensions important in the overview of the Rational Unified Process. First the time dimension, which consists of four phases and second static dimension, which is described in terms of activities, artifacts, workers and workflows (Rational Software, 2003). The phases of RUP are:

1. Inception
2. Elaboration
3. Construction
4. Transition

The workflows of RUP can be separated in process workflows and supporting workflows. The Process workflows are: Business modeling, Requirements, Analysis & Design, Implementation, Test and Deployment. The supporting workflows are: Configuration & Change management, Project management and Environment.

When these dimensions are combined Figure 5 is created (Kruchten, 2003). This model shows how the process is structured along two dimensions, it shows the workload on the different workflows during the iterations. Most of the requirements activity takes place in the inception phase and at the beginning of the elaboration phase.

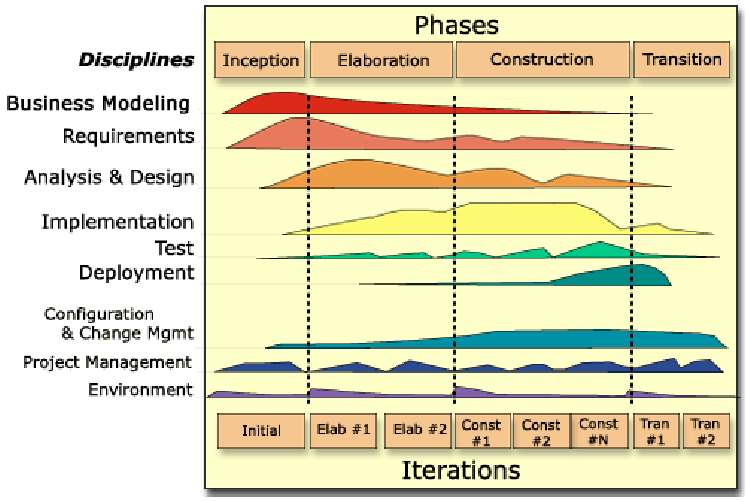

Figure 5: The RUP iterative model graph

The time dimension
The time dimension consists of the phases: inception, elaboration, construction and transition. Each phase concludes with a milestone. A milestone is a critical point in time where certain goals must be achieved and a critical decision must be made.

Inception; In the inception phase, the business case and the scope must be defined. The project is viewed from a high level. At the end of this phase the following artifacts must have been created: A vision document, an initial use-case model, an initial project glossary, an initial business case, an initial risk assessment, a project plan, a business model and one or several prototypes.

For requirements management the most important ones are: the vision document, the initial use case model and the initial business case to look at the business context and the success criteria. After this phase the first milestone is reached, the following evaluation criteria are reviewed before going to the next phase: Stakeholder concurrence, requirements understanding, credibility of the cost/schedule estimates, depth and breadth of the architectural prototype and actual expenditures versus planned expenditures. The project may be cancelled or considerably re-thought if it fails to pass this milestone.

Elaboration; The purpose of the elaboration phase is to deeply analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project. This is mostly considered the most critical phase of the project, this is where the decision is made whether or not to actually carry out the construction and transmission phases. At the end of this phase the following artifacts must be delivered: A use-case model (at least 80% complete), supplementary requirements, a software architecture description, an executable architectural prototype, a revised risk list, a revised business case, a development plan for the overall project, an updated development case specifying the process to be used and optionally a preliminary user manual.

For requirements management the most important ones are: the use case model, the supplementary requirements and the revised business case. After this phase the second milestone is reached, here the following evaluation criteria are reviewed before going to the next phase: Stable vision, stable architecture, major risk elements have been addressed, suffiecient plan for the construction phase, stakeholders agreement and acceptable resource expenditure versus planned expenditure. The project may be aborted or considerably re-thought if it fails to pass this milestone.

Construction; In essence the construction phase is the manufacturing and testing process. All application features are integrated into the product. At the end of the phase the product must consist of at least the following deliverables: The (integrated) software product, the user manuals and a description of the current release.

The only thing important for requirements management here is to see during testing if the software product matches the requirements. The third milestone is now reached, the evaluation criteria for going to the next phase are: product stability and maturity, stakeholders readiness for transition, actual resource expenditures versus planned expenditures. Transition may have to be postponed by one release if the project fails to reach this milestone.

Transition; In the transition phase the product is transferred from the developers to the user community. The product should be of acceptable quality to provide positive feedback, but usually issues arise that require new releases. The deliverables of this phase are: beta testing, parallel operation with a legacy system that it is replacing, conversion of operational databases, training and rolling-out the product to the marketing, distribution and sales teams.

The important aspect for requirements management is to see if the product meets the requirements. Instead of evaluation criteria the transition phase has objectives, these are: achieving user self-supportability, achieving stakeholder concurrence and achieving a final product baseline as rapidly and cost effectively as possible.

The static dimension

A process in general describes who is doing what, how and when. The Rational Unified Process describes this using the following elements:

Worker; A worker represent the “who” in the process. People tent to see one employee as one worker, but they should rather be viewed as roles or “hats” people can where. By doing so, one employee can perform multiple roles and thus can represent multiple workers.

Activity; Activities are the “how”. They are specific chunks of activities that a worker is asked to perform. An activity should be able to be included in the planning process. If the activity is to small it must be neglected, if it is to big it must be expressed in smaller parts.

Artifact; Artifacts represent the "what". An artifact is a piece of information created, used or modified by a worker or process. Artifact are the input for activities as well as the output of activities.

Workflow; Workflows represent the "when". Because a single activity doesn't make a process, they should be put in order. A workflow is a sequence of activities that produces a result of observable value. It can be represented as a diagram.

Core workflows

The Rational Unified Process has predefined nine core workflows, which represent a partitioning of the workers into logical groupings. As said, an employee can perform multiple roles, consequently employees can also be in multiple workflows. In repetition, there are six core engineering workflows and three core supporting workflows.

Core process workflows:

1. Business modeling workflow
2. Requirements workflow
3. Analysis & Design workflow
4. Implementation workflow
5. Test workflow
6. Deployment workflow

Core supporting workflows:

1. Project Management workflow
2. Configuration and Change Management workflow
3. Environment workflow

For the sake of this research, only the third core process workflow is explained, the requirements workflow. The goal of the requirements workflow is for the stakeholders and the developers to agree on what the system must do. The following are its key activities:

- Create a vision document, this document states the business goals and the needs of the stakeholders.
- Identify the actors, these actors represent not only the users, but also other systems that may interact.
- Identify the Use Cases, these use cases represent the behavior of the system. A complete overview is given in a Use Case Model using UML.
- Develop the Use Case Descriptions, show step by step what the system does and how it interacts with the actors.
- Specify non-functional requirements in the Supplementary Specifications

Advantages and disadvantages of RUP

The advantages and disadvantages of this approach are depicted in Table 1. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 1: Advantages and disadvantages of RUP

#	Description	Explanation
Advantages		
A1.1	Very complete	RUP has artifacts and guidelines for all the steps in the process of developing software.
A1.2	Proven solution	RUP has been proven to deliver software on time and on budget by a unified team in many traditional projects.
A1.3	Use of UML	RUP uses UML to model for instance Use Case Diagrams. UML is widely used and therefore understood by a lot of people.
A1.4	Clear requirements methods	RUP uses clear Requirements Management tools, artifacts and methods. Requirements are interwoven in the phases.
Disadvantages		
D1.1	Over complete	RUP consists of so many documents and guidelines that it can be overwhelming and someone could get lost in which ones to use.
D1.2	Prescriptive methodology	In practice RUP is seen as a very prescriptive methodology.
D1.3	Used as waterfall	RUP, although intended as iterative, is used as a waterfall approach.
D1.4	Phases are inflexible	RUP phases are not able to overlap, the milestones of each phase are very definitive and do not support any flexibility.
D1.5	No exploitation of BPMS abilities	RUP is not able to use the abilities of a BPMS, such as highly visual modeling, high flexibility and combined modeling and designing.

2.1.2 IRMA

The Integrated Requirements Management Approach (IRMA) is an approach developed by Capgemini to guide requirements management in Capgemini projects. There are a few versions of IRMA starting with what is called IRMA for RUP, which basically is the standard version. Then there are a few extensions. The first one is IRMA for Business Intelligence (IRMA for BI), which put more focus on for instance data modeling. This can be practical for BPMS projects and therefore the techniques will also be reviewed below. The second is IRMA for SAP. This focuses on Requirements Management in SAP projects only and contains guidelines and templates specifically for employees in SAP projects. Because SAP is not a BPMS, IRMA for SAP is considered out of scope for this research. The third extension is quite new and is called Smart Use Cases for IRMA, this topic will also be addressed. (Klabbers, Spier, Zijlstra, & Aalberts, 2011)

IRMA for RUP

As the name suggests IRMA for RUP is based on RUP, but adjusted for practical use within Capgemini. It not only recognizes the same phases as RUP, but also contains the RUP artifacts. For each of this artifacts, IRMA describes approaches, guides and templates (Klabbers, Spier, Zijlstra, & Aalberts, 2011) (Spier & Klabbers, 2010).

IRMA gives a more extensive list of artifacts of what they think is important for Requirement Management. For each artifact there is an extensive guide and template on how to create the artifact. In short IRMA describes the artifacts in the following way.

Vision; This captures the essence of the envisioned project. It gives an high-level overview of stakeholders and requirements. It's a fundamental document to communicate "what are we building?"and "why are we building it?" It sets the roles and rules for the stakeholders and gives clarity on the problem and the solution.

Supplementary Specification; This document captures the non-functional requirements. It takes the vision document as a starting point and lists the requirements on quality aspects such as usage, maintenance and design constraints. Next to non-functional requirements this document also contains generic functionality requirements. These describe requirements that do not belong to any specific use case, for instance: *When editing an item, there always has to be a cancel option, leaving the system in the state it was, before starting the edit action.*

Requirements Management Plan; This document describes how to gather the requirements, how to document, maintain and report them. It also describes how changing requirements are going to be handled in the project.

Software Development Plan; The Software Development Plan contains requirements on the software development process and the project deliverables.

Domain Model; In this model entities and their relations are shown. It creates a common vocabulary and describes the attributes and their responsibilities.

Glossary; The glossary defines the terminology specific to the problem. This ensures the reader isn't left with term unfamiliarity. The hard part here is to enclose enough terms, but not too many.

Use Case Model; This model consists of one or more Use Case Diagrams. It shows on a high-level which actors (users or other systems) are involved with the system and what actions they can perform. It is important that the model is simple and easy to understand for all stakeholders.

Use Case Specification; These are detailed descriptions of the interaction of an actor with the system that happens in a single unit of time, in a specific place. This is also abbreviated to OTOPOP, which means One Time, One Place, One Person. They describe basic flows, alternative flows and sub flows. The theory behind drawing up these use cases is based on (Zielczynski, 2008), (Eriksson, Penker, Lyons, & Fado, 2004) and (Cockburn, 2001). As (Cockburn, 2001) describes in his book, Use Cases can be specified very basic, giving just a description of an actor interacting with the system or they can be "fully dressed", describing also preconditions, scoping, triggers, guarantees, extensions, etc.
In this book he also describes different goal levels: A cloud level (very high summary of the goals, mostly the system name), a kite level (summary of the goals), the sea level (user goals), fish level (sub goals of the system), clam level (too low sub level). Everything above sea level are called white use cases and are actually too high level to be seen as one task. Everything under sea level are called indigo or even black use cases and are the sub division or the steps needed to complete a user goal. Exactly at sea level are the blue use cases which reflect the user goals. These correspond to the elementary business processes. The summary levels can go as high as the sky and the sub levels can go ocean deep, but there is only one sea level and therefore only one level for user goal use cases. The different levels are also shown in Figure 6.

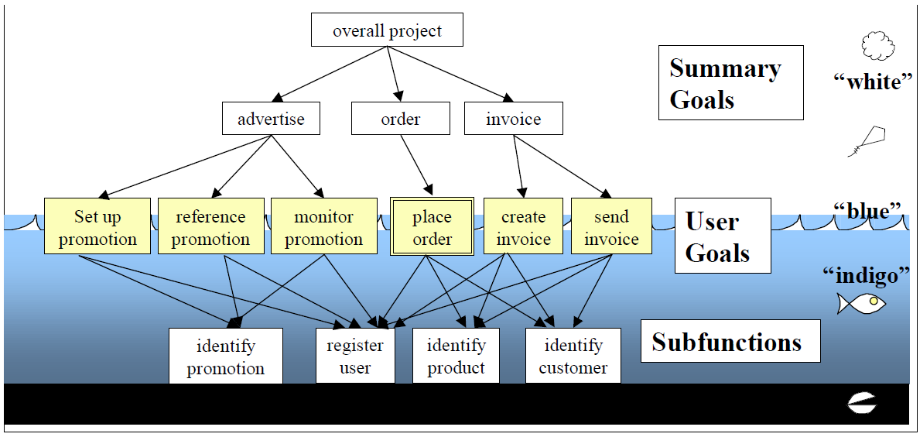

Figure 6: Goal levels of use cases (Cockburn, 2001)

System Rules; This contains all policies or conditions that must be satisfied before an action can be performed and that must be enforced by the system. It describes what the system must do when for instance a user has not enough authorization.

Service Definition; Describes the black boxes the system interacts with. The input is known, and the output is known. But not what happens inside the other service. The titles of these service definitions should be brief and clear, for instance: Request insurance proposal from Aegon.

Interface mapping; These mappings describe the same black boxes as the service definitions, but in addition they describe how the input and output data need to be transformed for both systems.

Storyboard; This is one of the two artifacts without a template, as it is a free format artifact. The goal of storyboards is to understand overall flow and interactions within a Use Case, not to prototype or test the look and feel of the user interface. The storyboard should not cover user-interface concerns. It is a sketch to validate user's expectations and their role within the use case. It could be anything ranging from Microsoft Word and Visio to screen shots and paper sketches.

Navigation map; This is the second artifact without a template. The navigation map is a visual representation of the manner in which a user may navigate between the various screens of the system. This is done to support accessibility of the system and stimulates reuse of user-interface elements. The navigation map is optional to use, but when used it has to be approved by all stakeholders.

IRMA for BI

IRMA for BI is based on IRMA for RUP is contains pretty much the same artifacts and it has the same elementary requirements types. The artifacts should contain the same information as described in the previous section with a few additions. For instance the vision artifact must contain a high level architecture and the requirements management plan has to contain ETL (Extraction, Transformation and Load) development standards. Furthermore there are two extra artifacts, these can be useful as a BPMS project often connects with multiple back-end systems:

Data Flow model; This model should describe how the data is transferred and transformed from the source systems to the demanded output. It is important to know that the dataflow model should be developed from the stakeholders point of view, not from the developers point of view. The data flows can be connected to the Use Cases.

Data Flow Specification; As the data flow model is high level, the data flow specification describes in more detail each data flow from the data flow model. Each data flow specification has three main elements: Input, the source to draw data from, output, the related target entity and the business rule, the transformation that the flow performs (the "how to").

Smart Use Cases for IRMA
Smart use cases describe the same functionality as regular use case descriptions, but in a different way. A way that is more visual and easier to communicate to the client. This method is more agile and requires more user participation. The best way to elicit these requirements is by holding a workshop with the end users and other stakeholders.

Smart Use Case modeling starts with building a Process Hierarchy, because it is sometimes difficult to start modeling processes right away it is best to brake them down into smaller steps. Start with the name of the application and then continue braking down until you reach the OTOPOP level. An example of a process hierarchy is shown in Figure 7. It must be possible to map each elementary use case one to one to a use case model or description.

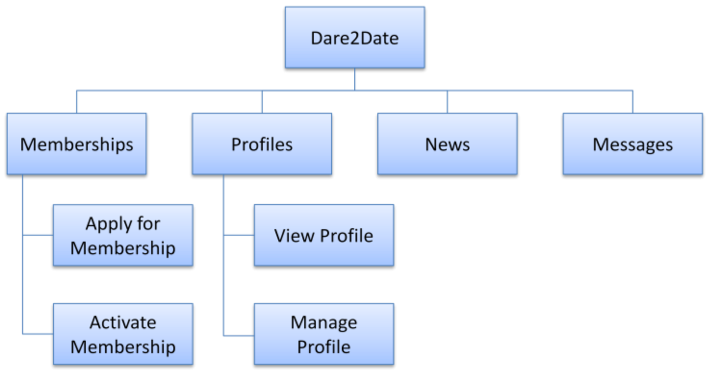

Figure 7: Example Process Hierarchy

The next step is drawing up the smart use case model. The bases of this model is the standard use case model also known in IRMA for RUP. The difference is that IRMA for RUP tries to capture all high level use cases and all actors in as less diagrams as possible, see for instance Figure 8.

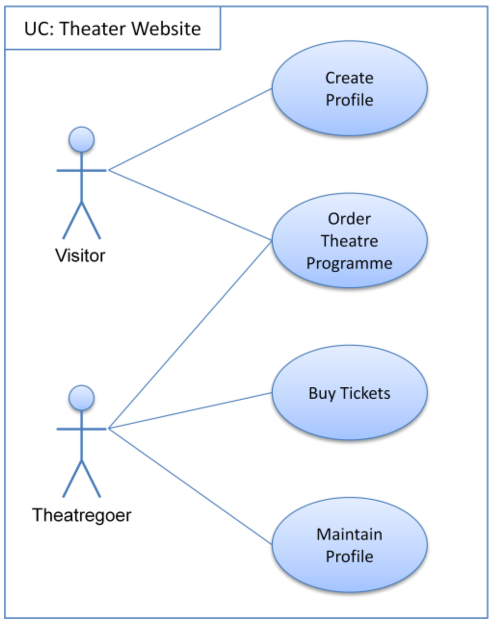

Figure 8: Example standard use case modeling

Smart use case modeling advices to use one model per use case and model it further by elaborating the sequence of operations that support this use case, thereby putting the functional complexity of a textual use case in a visual model. These use cases are then called sub-functional use cases or in the terms of (Cockburn, 2001) they are called the fish level use cases. An example of this is shown in Figure 9. Note that stereotypes are used to classify similar model elements and that there is an implicit order in the model. To read the Smart Use Case diagram always start with the main use case. Then continue clockwise with the sub-functional use cases starting from the top most one (indicated in Figure 9 by the red arrow).

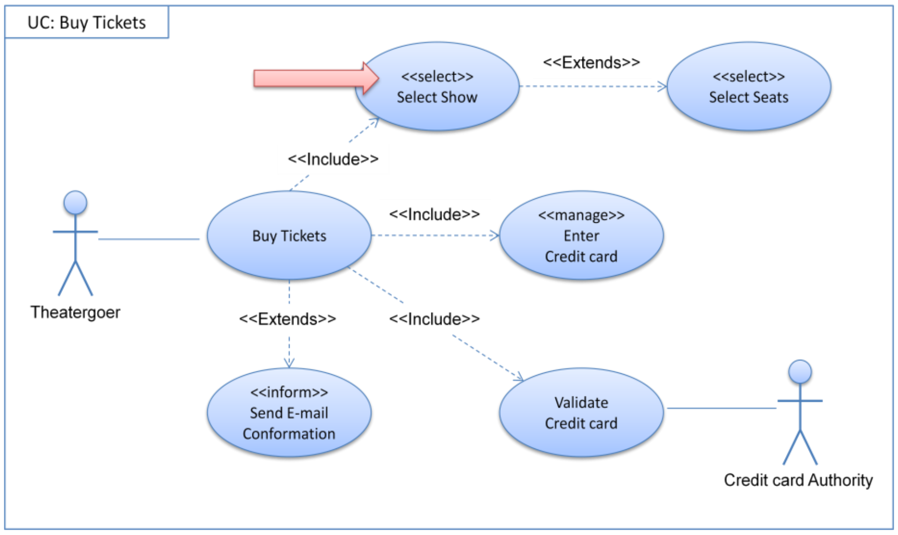

Figure 9: Example smart use case modeling

Advantages and disadvantages of IRMA

The advantages and disadvantages of this approach are depicted in Table 2. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

It might appear that advantages and disadvantages seem contradicting, this is intentional. For example, it's a good thing that there is lots of guidance and extensive templates, but it might be too much for BPMS use.

Table 2: Advantages and disadvantages of IRMA

#	Description	Explanation
Advantages		
A2.1	Specifically developed for Requirements Management	IRMA is specifically developed for Requirements Management and is not focused on other software development areas.
A2.2	Lots of guidance	IRMA offers a great number of standard templates, guidelines and quality procedures.
A2.3	Proven solutions	IRMA offers within Capgemini proven solutions based on worldwide proven solutions.
A2.4	Broad scope in Requirements	IRMA offers also specifications for BI solutions, which are more data oriented and Smart Use Cases, which are more visually oriented.
Disadvantages		
D2.1	Too many templates	IRMA offers maybe too many templates and guidelines to be used in a flexible process.

D2.2	Waterfall based	Just as RUP, IRMA is used in waterfall projects. It asks for lots of documentation and executive sign-offs. This does not support flexibility.
D2.3	Guidelines are highly regulatory	The guidelines itself are prescriptive. For instance, the use of Textual Use Cases is very specific, but can become complicated and may not be read by the stakeholders.
D2.4	No exploitation of BPMS abilities	IRMA is not designed to use the abilities of a BPMS, such as highly visual modeling, high flexibility and combined modeling and designing.

2.2 Agile approaches

2.2.1 AgileRUP

RUP is a very complete methodology. It contains more than 3500 files containing guides and templates for performing the activities and making the artifacts. Although RUP was originally intended to be a non-prescriptive and to have lots of possibilities to for instance scale down the various artifacts, lots of practitioners struggle with this idea and got the feeling that RUP still tells you what should be done by who exactly and when (Evans, 2006). That is why some practitioners tried to guide employees in making RUP Agile.

Agile RUP is the lighter version of RUP. It is not so much a different approach as it is a different mindset. The waterfall approach assumes that most (if not all) of a project is known upfront. That all requirements are clear before the system is designed. When looking at it this way, agile development, is much more humble, because it assumes you cannot know everything and surprises will come. The principles of an Agile Approach are (Beck, et al., 2001):

1. Individuals and interactions over processes and tools,
2. working software over comprehensive documentation,
3. customer collaboration over contract negotiation,
4. responding to change over following a plan.

This means developers have to stop working alone instead of together, stop designing before the problem is defined and stop seeking details to soon (Evans, 2006). To respond quickly to change and to quickly create working software short iterations are necessary. Because it is not possible to investigate all the requirements when working agile, the agile method suggest to start with the features of the system that generate the biggest benefit for the client with a relatively low effort, the so called low hanging fruit. For these features start eliciting the requirements, then design, build and test the system feature and then go to the next iteration.

One of the practitioners who learned to use RUP in an agile way is Michael Hirsch. In his paper (Hirsch, 2002) he describes not only what he changed about RUP for his projects, but also why he changed this and what went wrong anyway. Hirsch used RUP in an agile way at his own company around 1998. They adapted RUP but decided it was to make and needed to trim it down. Although every project is different the following alternations are suggested (their project had a team size of 4-7 people and had a project duration of less than nine months):

Artifacts; Maintain only artifacts that are really needed and that add value. Try to minimize the overhead. For the requirements phase in their project only the vision document and the use case model where used.

Activities; Use the activities mostly as a textbook rather than as finely grained workflow details. This is contrary to what RUP suggests, but worked well in their case, because the development team was small and all developers were experienced.

Roles; Assign no formal roles, but rather us them as checklist to verify that all the required skills are there.

Project Planning; Base the project plan on results to achieve rather than plans based on a list of tasks to be done.

Phases; Keep all phases of RUP

Iterations; Use iterations of about four weeks. Each resulting in a tested software release.

Project Control; Use weekly status meetings with the entire project team. Establish per person what he or she has done and what still needs to be done to achieve his goal. These responsibilities and statuses must be very clear.

Capgemini Agile RUP

Capgemini developed their own Agile RUP method. Capgemini based Agile RUP on Rational Software's RUP, its light-weight open-source cousin OpenUP (Open Unified Process) and additional agile practices derived from other agile methods such as Scrum and eXtreme Programming (Capgemini, 2011c). They kept the phases from RUP: inception, elaboration, construction and transition.

Within these phases a scrum aligned approach is chosen. From the elaboration phase onwards the result of each iteration must be the demonstration of executable software. The feedback from the users on this demo should be considered as requirements for the next release. A product backlog is created (a prioritized to-do list for the project). Then with each iteration a so called 'sprint' back-log is created for the team to work on for the next 2-4 weeks to create the next production ready release. More on Scrum will be explained in section 2.2.3 Scrum. When looking at the phases of agile RUP and the role of requirements in this method, these are the key features:

Inception; In the inception phase the project scope is defined, the environment is established, the high-level plans are validated and the high level design is determined. For the requirements engineer or business analyst it is important to develop an understanding of the required functionality of the system. In collaboration with the client it is very important to prioritize and validate requirements and determine dependencies between them.

Elaboration; In the elaboration phase the objectives are to: mitigate the risks, prove the chosen software architecture will support the critical functional scenarios, demonstrate working application components and validate high-level estimates. For the requirements engineer it is important to elaborate on the requirements to develop a more detailed understanding of the functionality on an iteration-by-iteration basis. The client must be there to cooperate and act as a business stakeholder, a question resolver, a subject matter expert and a decision maker in this process.

Construction; The construction phase in this matter looks quite similar to the elaboration phase. Requirements still need to be elaborated, but the scope is now more on construction. The iterations contain more detail. The role of the client is also pretty much the same.

Transition; The role of requirements is very small in this phase. The only thing to keep in mind is that the result of the acceptance tests are new requirements for the next iteration of the continuous development. Because there are more iterations with agile RUP, requirements will also come more frequent.

Capgemini Agile RUP furthermore specifies a long list of requirements deliverables that can be created when doing an agile RUP project. It is important to know that some artifacts are recommended, while most of them are only to be produced if it's required by the project circumstances. The complete list of requirement deliverables according to Capgemini Agile RUP can be found in Table 3 (Capgemini, 2011c):

Table 3: Requirements deliverables Agile RUP

Artifact	Recommended / Optional	Description
Activity Diagrams	Optional	Representation of functional process
Backlog	Recommended	Prioritized list of requirements
Business Rule Catalogue	Optional	Rules the system should follow
End to end scenarios	Optional	The most frequent operational scenarios that users face
Glossary	Optional	Important business terms
Interface Specifications	Optional	Details of the data contained in an interface message
Logical Domain Model	Optional	Domain objects, their attributes and their relationships
Message Catalogue	Recommended	Errors, warnings or information messages from UCs
Outline Use Case Specifications	Recommended	Details the functionality to be performed
Requirements Management Process Description	Recommended	How requirements will be gathered, documented and managed
Report Specifications	Optional	Detail/layout of the reports that are produced within UC
System Wide Requirements	Recommended	Non-functional and functional requirements that are not covered in UCs
Traceability Report	Recommended	Tracing between the requirements and project deliverables
Use Case Diagram	Optional, but recommended	System's intended functions and its surroundings
Use Case List	Recommended	Textual description of the Use Cases

Use Case Modules	Recommended	Details an individual scenario of an Outline UC Specification
Use Case Storyboards	Recommended	Document the screens that are required and the navigation between those screens
Vision	Recommended	High-level requirements and design constraints

Advantages and disadvantages of Agile RUP

The advantages and disadvantages of this approach are depicted in Table 4. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 4: Advantages and disadvantages of Agile RUP

#	Description	Explanation
Advantages		
A3.1	Light weight RUP	Agile RUP is a light weight version of RUP with less prescriptive artifacts. Some are recommended, but not mandatory.
A3.2	More humble	Agile RUP is a more humble approach. This means it assumes you cannot know everything in advance.
A3.3	Iterative and incremental within phases	Agile RUP uses an iterative and incremental approach within the phases, which leads to flexibility as well as structure.
A3.4	Hands-on deliverables	The deliverables of Agile RUP are detailed and use a hands-on approach.
Disadvantages		
D3.1	Not that light-weight	Although Agile RUP is a light weight version it still has a lot of artifacts that are recommended to deliver. Maybe too many for BPMS project use.
D3.2	Inconsistent definition of Agile RUP	No real consensus on what Agile RUP is. Some versions are more structured than others and some have adapted the ‘agile mindset’ more. This leads to inconsistencies amongst authors in for example how mandatory the artifacts are or on what project sizes this is applicable.

2.2.2 OpenUP

OpenUp is a standard derived from Rational Software's RUP by the Eclipse Foundation. OpenUP has, just like RUP, a very complete knowledge base and a very extensive library of methods and techniques (The Eclipse Foundation, 2010). The two differences are that first of all, OpenUP is open, so it is free to use for everyone. Second, OpenUP adapts a more agile way of working. OpenUP contains the minimal set of practices to help teams be more effective in developing software. Being lightweight is the first reason for claiming agility. The second one is that OpenUp uses an incremental and iterative approach (Balduino, 2007).

OpenUP has four core principles that capture the general intensions behind a process and create a foundation for interpretation of the different aspects of OpenUP. The four core principles are:

1. Collaborate to align and share understanding
2. Balance competing priorities to maximize stakeholder value
3. Focus on the architecture early to minimize risks and organize development
4. Evolve to continuously obtain feedback and improve

The process

A project performed using OpenUP consists of the four phases of RUP. Inception, elaboration, construction and transition. The combination of the four phases is called the project lifecycle and is captured in a project plan. Within these phases the project is developed in time-boxed iterations, the activities for such an operation are captures in an activity plan. At the end of an iteration an demo-able build is delivered. Every day is called a micro-increment, where people work on a work item. This way of working has similarities with the Scrum methodology. The difference is that these iterations are within a certain phase, whereas Scrum is multidisciplinary. A graphical representation of the OpenUP way of working is shown in Figure 10.

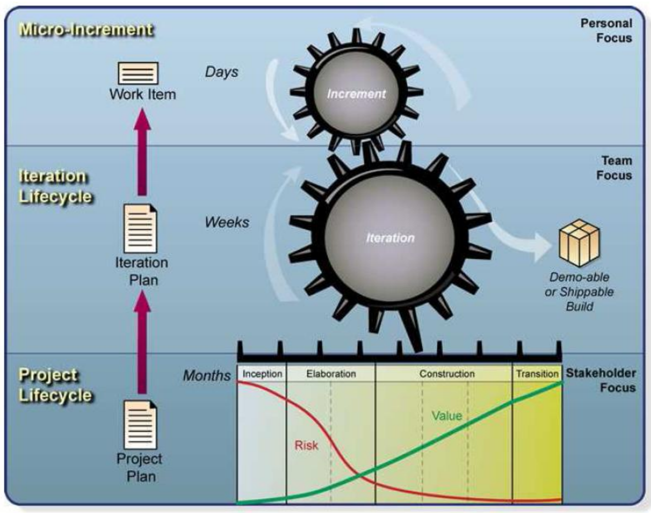

Figure 10: Organization of OpenUP

Roles
Roles are the skills needed in a development OpenUP Team. The team is mostly small and co-located. The roles or skills needed in a team are: Stakeholder, analyst, architect, developer, tester, project manager and the any-role, which is anyone on the team that can perform general tasks (Balduino, 2007). These roles are very basic and traditional roles. Their responsibilities are also nothing new.

Disciplines and work products
The OpenUP method exists of a few disciplines (Balduino, 2007). These disciplines are: Requirements, architecture, development, test, project management and configuration & change management. Each of these disciplines come with a few work products. Work products are artifacts that need to be realized by the roles responsible for this discipline. For the requirements discipline the following work products need to be produced: Glossary, Vision, System Wide Requirements, Use Case Model and Textual Use Cases.

Advantages and disadvantages of OpenUP
The advantages and disadvantages of this approach are depicted in Table 5. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 5: Advantages and disadvantages of OpenUP

#	Description	Explanation
Advantages		
A4.1	Open standard	OpenUP is, as the name suggests, is an open standard and documentation is available for anyone.
A4.2	Lots of Guidance	OpenUP provides a lot of artifact, tools and guideline documentation.
A4.3	Iterative and incremental within phases	The process of OpenUP is a combination of an iterative and incremental approach combined with the phases of RUP, which provides structure and flexibility.
A4.4	Proven Solution	OpenUP is a widely used standard and has proven itself in practice.
Disadvantages		
D4.1	Can be too much information	OpenUP has so much documentation that users can be overwhelmed or get lost in the information.
D4.2	Work products not complete	The set of work products delivered is very concise and is only a bases set of deliverables.
D4.3	Roles are stubborn	The roles of OpenUP are not flexible. This doesn't comply with the agile or scrum like project approach it suggests.

2.2.3 Scrum

Scrum is not so much a project management method or a requirements management method as it is a way of thinking. Where the methods sometimes seem similar to other approaches it must be emphasized that the beliefs of scum are much more strict and are deeply inside developers who really have adapted scrum. Although it is most of the time not directly linked by people to requirements management, is it has some interesting ideas and characteristics. Scrum uses an iterative and incremental approach to increase predictability and reduce risk. A visual model of these iterations is shown in Figure 11. As a foundation, scrum has three pillars of empirical process control, which guide people through the process (Schwaber & Sutherland, 2010):

1. Transparency; The effects of a process or iteration must be very visible. Also the definition of when something is finished or "done" must be clear.
2. Inspection; The process must be inspected frequently to ensure variances and risks can be detected early.
3. Adaption; If it is noticed during inspection that some aspects are outside tolerable limits, adjustments must be made quickly.

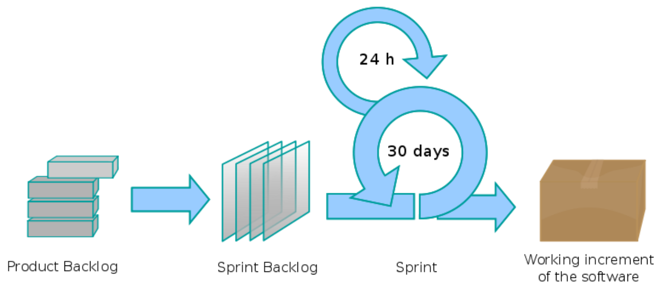

Figure 11: Scrum process

The scrum framework exists of four main aspects. These are: Roles, Time Boxes, Artifacts and Rules (Schwaber & Sutherland, 2010). The first three are described below, the rules is what binds them all together.

Roles

Every scrum project is organized to be flexible and productive. The teams therefore should be multidisciplinary and self-organizing. Every team knows three roles, no more, no less. The first one is the Scrum Master, who should take care that the process is understood and followed. The second one is the Product Owner, who has the responsibility over the overall product and should maximize the value of what the scrum team does, most of the time this role is performed by someone of the client. He controls the product backlog, which is the prioritized list of requirements. The third and final role is the development team, which exist of seven, plus or minus two people that actually carry out the work. The Team consists of developers with all the skills to turn the Product Owner's requirements into a potentially releasable piece of the product by the end of the Sprint. There are no formal roles like architects or testers within the team, but all the skills should be there.

Time boxes
Several elements of scrum have to be performed within a time-box to create regularity. These include: the release planning meeting, the sprint planning meeting, the sprint, the daily scrum , the sprint review, and the sprint retrospective. The heart of every scrum is the sprint. This is an iterations of a preset length (mostly a month), in which an increment of the final product is delivered. When one sprint ends, the next one immediately begins.

In the release planning meeting the overall planning and goals of a project are stated. It is kind of like a vision document. The difference is that this document starts out very high level and is adjusted every time at the beginning of a sprint. The basic idea is that you cannot now your complete vision at the start of a project. In the sprint planning meeting the development team establishes “what” is build during a sprint and “how” the team will do this. From the product backlog the items with the highest priority are transferred to the sprint backlog.

At the end of a sprint there is a sprint review where the content and the achievements of the sprint are discussed. What went right, what went wrong, which problems were solved and a demo of the deliverable is shown. Adjacent there is a sprint retrospective where the process is reviewed. The team discusses how to improve the use of the Scrum tools and methodologies to improve the process. Last but not least there are daily scrums of maximum 15 minutes. These are important to update the team members on each other’s status, achievements and next challenges.

Artifacts
There are four artifacts involved with scrum. The product backlog, the release burndown, the sprint backlog and the sprint burndown.

The product backlog as mentioned before is a list of requirements of the product as a whole (not per sprint). They can be in the form of use cases, but consist mostly of user stories. An important thing to remember when developing with Scrum is that the product backlog is never complete. The first version only contains the most important and most known requirements. The product backlog is only complete when the product doesn’t exist anymore. The product backlog is prioritized by the team and the product owner. The product owner is in charge of the product backlog. Decisions about the product backlog can only be taken by the product owner (although he can be advised by others) and the team has to respect these decisions. The release burn down is the sum of the estimations about the amount of work that is left. The unit of the total is mostly in sprints.

The sprint backlog is derived from the product backlog during the sprint planning meeting. It consists of the tasks to be performed by the team that fulfill the requirements with the highest priority. The requirements are split up in smaller tasks that can be performed within one day. The sprint backlog can be altered during the sprint. The sprint burndown is a graphical representation of the total amount of work left plotted against the available time left.

Done
Two important things remain. First, the definition of done. It can be (it mostly is) that different team members have different interpretations of when a deliverable is

considered done. It is important to align this along the team members and with the product owner. Second, what to do with things that haven't been finished within a sprint. These things go on a separate list of undone work, which is then added to the product backlog. This way the release burndown stays reliable.

Advantages and disadvantages of Scrum

The advantages and disadvantages of this approach are depicted in Table 6. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 6: Advantages and disadvantages of Scrum

#	Description	Explanation
Advantages		
A5.1	Transparency	Scrum is a transparent approach that focuses on communication and openness between the team members.
A5.2	Adaptability	Scrum states that you cannot know everything in advance and you should adapt to changes that come on your path along the way.
A5.3	Fast working software	Scrum makes sure working parts of the product are delivered from the first iteration.
A5.4	Team responsibility	Members of the development team are self organized and own more responsibility.
Disadvantages		
D5.1	Full adoption needed	Full adoption of the Scrum process is needed for it to succeed. A semi-adapted Scrum approach or 'Scrum In Name Only' approach is likely to fail, due to mixed expectations.
D5.2	Full commitment from client needed	Full commitment by the client is needed. If the client or executive board doesn't fully trust the Scrum team to develop software in this self-organized way, the approach is likely to fail.
D5.3	Experience needed	Experience within a Scrum team is needed to make them self-organizing and be able to create less documentation, also in requirements.

2.2.4 Visual requirements modeling

What is meant by visual requirements modeling is to elicit, document and manage requirements using only graphical deliverables. Two techniques are discussed.

Business process based requirements modeling and management

(Pichler & Rumetshofer, 2006) discuss in their paper a new requirements elicitation and management technique they used in a real-life case using visual models. It is based on the idea that users do not know exactly what they want, but they'll know it when they see it.

They stress the importance of visual modeling as *"a meaningful way to support a mutual understanding among stakeholders in a project"*. As a picture says more than a thousand words, visualization improves communication between customers, users and development teams. Users sometimes need a little education on modeling notation, but they think it is interesting and positive to visualize their work processes and understand the meaning of used elements very fast.

They developed a Business process-based requirements engineering model, shown in Figure 12. The model consists of four steps: Client's vision, Analysis, Design and Implementation/Test. Each step exploring more and more of the depths of the customer's point of view, to create understanding of the to-be business process. The workshops used in these steps were dedicated to deliver one particular result.

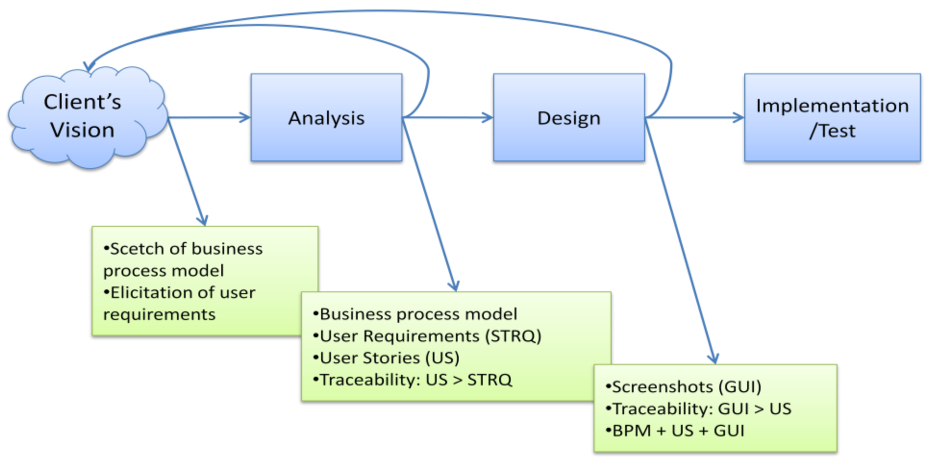

Figure 12: Business process-based requirements engineering

Rapid Design and Visualization

The visual requirements modeling technique used at Capgemini is Rapid Design and Visualization (RDV). This is a method based that uses the tool iRise, but can also be used in combination with other tools like Microsoft Visio and Adobe Photoshop. Almost everything that can draw up fast visualizations needed for the project. The main focus is on building prototypes and involving the stakeholders and end users to test, evaluate and collaborate in an interactive way.

RDV is based on User Centric Design (UDC), which was first introduced by (Gould & Lewis, 1985), where they introduced three principles: An early Focus on users and tasks, Empirical Measurement and Iterative Design. In essence User Centric Design is *"a broad term to describe design processes in which end-users influence how a design takes shape."* (Abras, Maloney-Krichmar, & Preece, 2004). To come to visualizations RDV takes the following steps:

1. Observe the users, using recordings, notes and observations.
2. Document these findings
3. Generate the user requirements from these observations
4. Develop design insights, considering user pain points and standards.
5. Create visualizations and build prototypes.

Advantages and disadvantages of Visual Requirements Modeling

The advantages and disadvantages of this approach are depicted in Table 7. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 7: Advantages and disadvantages of Visual Requirements Modeling

#	Description	Explanation
Advantages		
A6.1	Fast requirements development	Creating visual models is much faster than describing everything in plain text.
A6.2	Encourages discussion	Visual models are more attractive to business people and are faster understood.
A6.3	Helps users know what they want	Building visualizations and prototypes gives users an idea of what the product will look like, this is an easy opportunity for feedback.
Disadvantages		
D6.1	Not detailed enough	Although visual models are a valuable addition, detailed requirements or technical issues cannot be documented this way. When using a BPMS to connect to back-end systems for instance.
D6.2	Business Processes not always visually complex	A process can be very straightforward, regarding user interface. Especially when using the BPMS standard tools for building forms and interfaces.
D6.3	Room for interpretation	Visual models aren't as unambiguous as text can be. A wrong picture can say a 1000 wrong words.

2.3 BPMS vendor approaches

There are a lot of Business Process Management Suite vendors today. Because of time limitations these cannot all be reviewed for their methods on requirements management. As scope, three major vendors are chosen. This selection is based on the availability of expertise of these systems within Capgemini as well as their leading position in the Gartner Magic Quadrant for BPM Suites (see Figure 13) (Gartner Inc., 2010). Vendors upholding a leading position are more likely to have complete and widely applicable methods for requirements management.

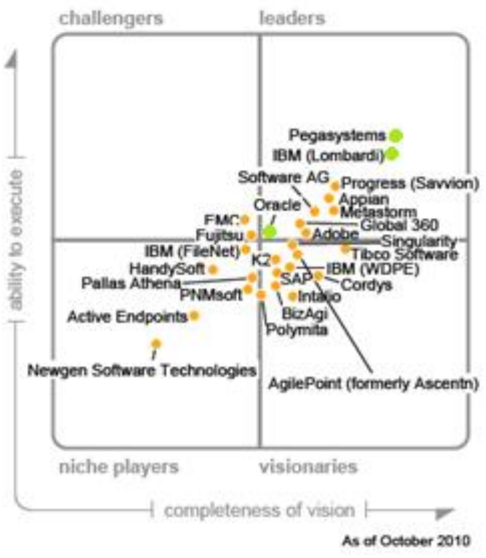

Figure 13: Gartner Magic Quadrant for BPM Suites (Gartner Inc., 2010)

The selected vendors are: Pegasystems, IBM BPM (also called Lombardi) and Oracle BPM. The sections below do not describe how to operate the BPMSs, but explain the methods and approaches prescribed by the vendors to manage a BPMS project, thereby focusing on requirements management.

2.3.1 Pega

The methodology Pega developed for developing software using their BPMS is called SmartBPM (Pega Developer Network, 2011). SmartBPM is based on the Rational Unified Process and consists of the following phases, which are quite similar to the ones of the RUP. There are two main differences:

1. Project initiation and go-live are acknowledged as actual phases
2. The elaboration and construction phase are combined in an intertwined iterative phase.

The model is shown in Figure 14.

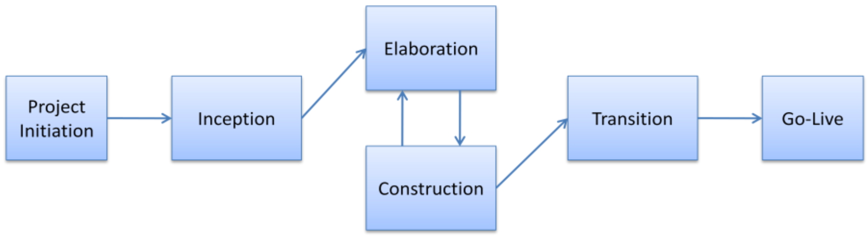

Figure 14: Pega SmartBPM phases

Project Initiation

The project initiation is the start of the project. The timeframe for this ranges somewhere from a week to a maximum of six months. The main activity is to exchange knowledge. From the client to the developer, but also from the developer to the client. Together with the project team of the developer the client walks through the as-is process. This way the current state is reviewed and understood by the project team and initial plans for process improvements can be made. Developers should inform the client on the process to follow, proof of concepts and product demos. This communication mostly happens in a structured workshop where the whole project team including the client meet face-to-face. In these workshop(s) the projects program is developed. Milestones are identified, roadmaps are setup and the project is divided into smaller projects and project slivers. A sliver is a combination of all activities that are needed to bring a piece of work to production. The roles involved in this phase are the business sponsor, the project manager, subject matter experts and supporting technical resources.

Inception

This phase literally stands between the initiation of a project and the elaboration and construction phase. The primary goals have been set and now it is time to make the project more concrete. Identify which artifacts, processes, tools, and resources will be used in this phase. The slivers are organized and ordered by business impact and effort, so that quick wins can be identified. This phase will in the end deliver an

Application Profile. The Application Profile is a scoping document, that is generated using the Pega tool. It contains the following information:

- **Actors**; States which roles to be modeled in the process.
- **Requirements**; Describes exactly what the business is looking to build. Mostly in the form "the system must provide…". These are then linked to one or more use cases. Use cases are still high level during the inception phase. They are detailed later.
- **Work Types**; Describe focus points for the business. They are used to classify requirements.
- **Interfaces**; State what screens should be visible for the user and what they will generally look like.
- **Reporting**; Specifies what reports need to be generated for managers.
- **Correspondences**; Specifies what correspondences exist between the process flow and external systems.
- **Assumptions**; Standard assumptions can be generated as well as own additional assumptions about the project.
- **Participants**; States which participants there are during the development of the system. For instance: "project manager".
- **Sizing**; Based on the previous information and modifications made by experts, the Pega sizing tool gives an estimation of the duration of the project.

Elaboration & Construction

In the elaboration and construction phases the details of development are worked out and then build. This happens in an iterative way. Using intensive workshop with the client, called Direct Capture of Objectives (DCO) sessions, details about requirements, use cases and work types are gathered. The BPMS is used in these workshops to demonstrate the processes as well as prototypes and mock-ups of the software. These DCO sessions take place every cycle and focus on only one work type with several use cases. Using white-boards and the Pega tools the high level process flow is drawn. A DCO session defines the following roles: Meeting moderator, scribe, business architects, system architects, process owners, subject matter experts/business analysts, IT representatives, testing representatives and training representatives. A DCO session is focused on collaboration and is a forum for clients, business analysts, testers and trainers to make sure that a minimum of inconsistencies exists between the different parties involved. Nothing should be developed as a silo. At the end of the DCO sessions all the elements described in Figure 15 must be clear and contain no inconsistencies (Pega Developer Network, 2010).

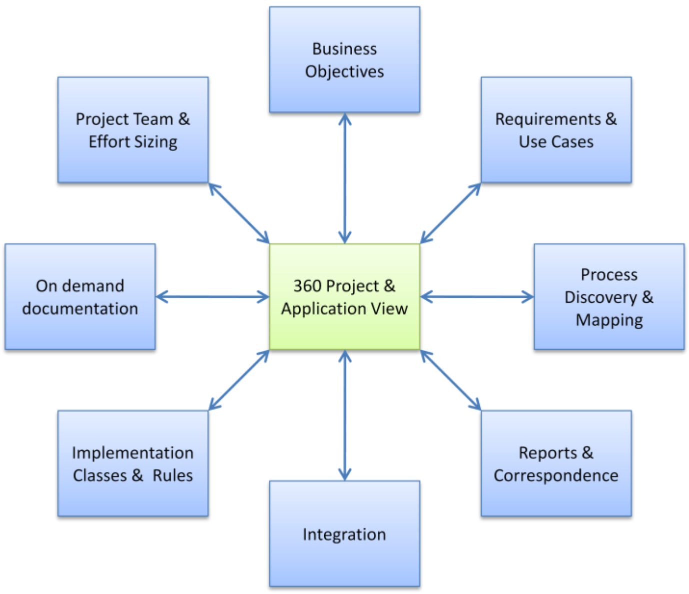
Business Objectives
Project Team & Effort Sizing
Requirements & Use Cases
On demand documentation
360 Project & Application View
Process Discovery & Mapping
Implementation Classes & Rules
Reports & Correspondence
Integration

Figure 15: DCO end result

Transition
The transition phase ensures that the finished application is of an acceptable quality. The phase includes the user acceptance tests and guides the transition from one environment to the other.

Go-Live;
The Go-live phase takes the final release candidate from the transition phase and supports its launch. Furthermore the application is continuously maintained with enhancements, fixes and change request from business as well as IT users.

Advantages and disadvantages of the Pega methodology
The advantages and disadvantages of this approach are depicted in Table 8. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 8: Advantages and disadvantages of the Pega methodology

#	Description	Explanation
Advantages		
A7.1	Uses BPMS possibilities	SmartBPM is an approach specific for BPMS projects (using Pega). It therefore makes use of the possibilities a BPMS offers.

A7.2	Just In Time detailed requirements	SmartBPM has an iterative and incremental Elaboration and Construction phase to gather Just In Time detailed requirements.
A7.3	Artifact tooling	SmartBPM offers tooling to create artifacts.
A7.4	DCO sessions	SmartBPM uses DCO workshops extensively to exchange the necessary information with the client on a regular bases.
A7.5	Expectation management	SmartBPM focuses on a high customer involvement to communicate expectations.
Disadvantages		
D7.1	Pega specific parts	Parts of SmartBPM are not applicable to projects that don’t use Pega as a BPMS.
D7.2	Tooling not flexible	SmartBPM using the tool does not offer the flexibility to create steps in the artifacts differently, for instance use cases in the Application Profile.
D7.3	No requirements data modeling	Modeling or documentation of requirements data models is not included in the SmartBPM method. This is considered to be captured in the flows and screens.

2.3.2 IBM Business Process Manager

For the BPMS of IBM, IBM BPM, two important documents explain how the process of developing a BPMS takes place. The first is (IBM, 2009). This prescriptive guide shows a brief step-by-step, day-by-day plan for implementing a BPMS from scratch in sixty days, in a divisional scenario, not organization wide. The second is (Bergland, Maquil, Nguyen, & Son, 2009). This document elaborates on the first one, explaining the concepts and giving a practical examples of a scenarios where a BPMS is implemented. Both state that for implementing IBM BPM there are five phases: Discover, Storyboard, Experience, Manage and Deploy. The first four are iterative. When those four are completed the process moves to the deploy phase. This is also shown in Figure 16.

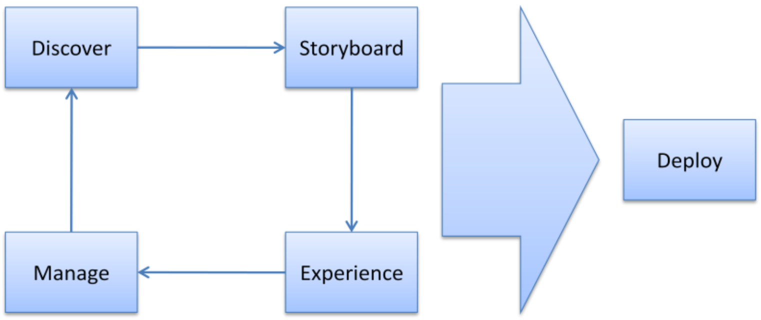

Figure 16: Phases IBM BPM

Before describing these phases in more detail it is important to know the different roles of this approach. The IBM BPM approach describes six primary roles. Three business roles and three IT roles.

The business roles are business analyst, subject matter expert and business executive. For requirements the business analyst is important. This is someone with a business background, not a technical background. It is someone who has been with the company for several years and has extensive knowledge about the business and IT goals. It should mostly be senior personnel who is able to identify process improvements.

The IT roles are solution developer, IT administrator and IT architect. The IT architect analyzes and prioritizes the requirements. This person is responsible for designing the requirements for the interfaces with other systems. The architect must have detailed understanding of the overall technical goals and the IT landscape of the organization.

Most of the work the IT architect does, most deliverables he creates cannot be created directly in the BPMS. A separate tool called IBM Blueprint is used that can be integrated with the BPMS.

Discover

The goals of the discover phase are: to identify the business challenges, define goals to meet those challenges and define business measures to know if those challenges are tackled. To achieve this, the discover phase has the following deliverables:

Strategy map; A strategy map is a high level view that starts with the business objectives. What bottlenecks can be tackled, what are the goals of this system. In the same map is sketched how the solution (IT system) can help reach these goals. It illustrates the strengths and the weaknesses of the environment to implement this solution. Last but not least this map shows the business measures for this project. These can be for instance Service Level Agreements (SLAs) or Key Performance Indicators (KPIs)

Business Capabilities map; The business capability map ensures the organizational capabilities align with the strategic objectives. This helps identify gaps between what the company wants to achieve and what can be achieved. It contains mostly a mapping of organizational resources and IT resources against the business objectives. It is possible to make a high level capability map with the possibility to drill down.

Process maps; These maps show the high level version of the process models to reach the business goals, aligned with the business's capabilities. It is possible to drill down within these models to create a little more detail in the process, but important to remember is that is should always remain high level. Technical implementation issues should not be in these maps.

Storyboard

In the storyboard phase the goal is to develop the requirements for the future process. It starts from analyzing the current processes. By modeling these in BPMN (Business Process Modeling Notation) in the IBM business modeling tool and then simulating the process it's possible to determine the most expensive and the less efficient paths in the process. Using this analysis, improvements can be suggested and a future state models can be defined by creating business rules and mock-up forms. New KPIs can be determined. The storyboard phase has the following deliverables:

Current state process model; At the beginning of a project the current state process is probably in paper form, some high level diagram or just in the heads of the employees. The trick is to model it in a standard form (BPMN) to be able to analyze it using IBM BPM. When simulating this process using the tool, the less efficient paths are determined.

Future state process model; By improving the bottlenecks in the current state process model, the future state process model is created. It demonstrates where the business can gain value. The creation of a future state process can take multiple iterations.

Mockup forms; Mockup forms are simple forms to show the end user what the screens will look like. This helps to validate the process. Validated mockup forms are needed to go to the next phase.

Experience
In any other phase system this would have been called the development phase or something similar. In this case it is called the experience phase. This means the models from the storyboard phase are elaborated, they are further developed through experiencing how they work, experience the solution.

Manage
The manage phase assumes that the KPIs of the project have been identified during the storyboarding phase and refined during the experience phase. These KPIs are set as measures in the manage phase. This way bottlenecks can be identified, which can lead to the optimization of work assignments.

Deploy
The deployment phase is the only phase which is not involved in the iterative model as shown in Figure 16. The IBM BPM guide describes that the first four phases are business oriented. There is a possibility for managers and business analyst to review processes and KPIs and iteratively fine-tune them. Now the focus is on the IT team. The goals of this phase are to design a solution architecture and setup the environment, prepare and deploy solution artifacts, unit test the solution and eventually keep monitoring the health of the solution.

Advantages and disadvantages of the IBM BPM methodology
The advantages and disadvantages of this approach are depicted in Table 9. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 9: Advantages and disadvantages of the IBM BPM methodology

#	Description	Explanation
Advantages		
A8.1	BPMS specific approach	IBM BPM uses an approach specific for BPMS projects (using IBM). It therefore makes use of the possibilities a BPMS offers.
A8.2	Use of KPIs	The IBM BPM methodology focuses on creating measures and KPI with each requirement and adjust them during the phases when needed.
A8.3	High amount of visual artifacts	Only visual artifacts are used to create strategy maps, business capability maps, process maps, process models and mockup forms.
Disadvantages		
D8.1	No focus on requirements artifacts	There is no focus on creating specific requirements artifacts, these are combined with process flows and strategy maps.
D8.2	No iterations from deployment phase	New requirements emerging from deployment cannot create a feedback loop to the discover phase in the model.

D8.3	Requirements stay high-level	Requirements stay at a very high-level, there are no detailed descriptions of requirements. Neither does the approach give the guidance or templates to do so.
D8.4	No requirements data modeling	Modeling or documentation of requirements data models is not included in the IBM method. This is considered to be captured in the flows and mockup forms.

2.3.3 Oracle BPM

The Oracle Business Process Management Suite consists of many tools and applications, which its method is build around. Who uses these tools and when do they use them? The answer to “who” is given in the description of the roles and the answer to “when” is given by the application development lifecycle (Oracle, 2010).

Roles

The Oracle BPM Suite knows five different roles, they are types of users (also called User Personas). They have different responsibilities, are involved in different stages of the development cycle and use different components of the Oracle BPMS to perform their job. A description is given below.

Process Analyst; The job of the process analyst is to create the initial business process flow, identifying the initial Key Performance Indicators (KPIs) and perform simulations to estimate the Return on Investment (ROI). Process analyst typically use the Oracle Business Process Analysis (BPA) Suite or Business Process Composer to create process models. They may also use the Process Analyst role within Oracle BPM Studio. The components of Oracle are explained at the development lifecycle section.

Process Developer; Their job is to implement the process models created by the process analyst. This could be an internal process step or communication with a back-end application. Process developers typically use Oracle BPM Studio to model and implement the components. They may occasionally use Business Process Composer for modeling basic processes.

Business Administrator; Business administrators are responsible for administering the BPM infrastructure. This includes installing and setting up the BPM environments and governs BPM hosting. The main tool for them to use is the Oracle Enterprise Manager.

Process Owner; The process owners can be compared to a project managers. They are responsible for controlling and managing the development process. They analyze the current state of the process using dashboards and other metric analysis tools. Process owners typically use Oracle BPM WorkSpace. They may also use the Oracle BAM console to view the dashboards.

Process Participant; Process participants can also be described as the end user. They are the ones working with the application created by Oracle BPMS. The systems they use are use Oracle BPM WorkSpace or Process Spaces.

Application development Lifecycle

Just like many other methods the Oracle BPMS implementation method has a four phase lifecycle. The phases that are used are: Modeling, Implementation, Deployment and Run-time. Figure 17 depicts not only which roles are involved in what phases, but also what Oracle BPM tools and applications are used and the process interaction between them.

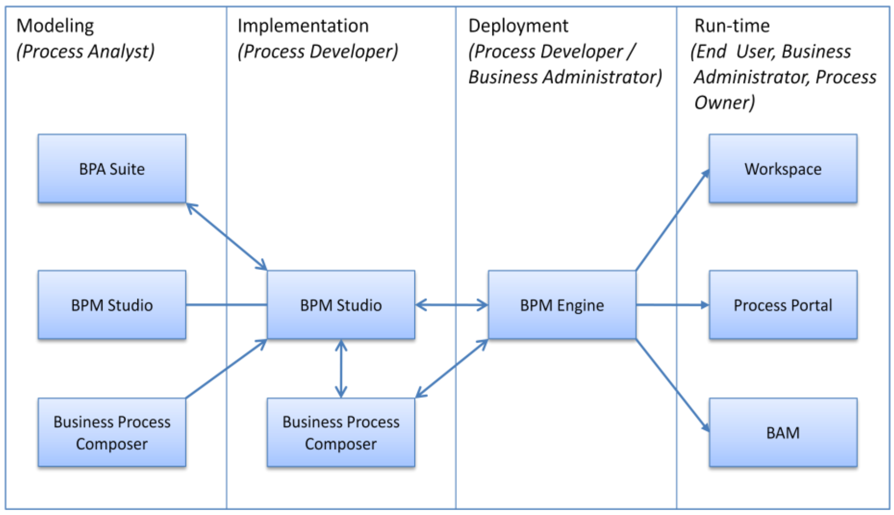
Modeling
(Process Analyst)
Implementation
(Process Developer)
Deployment
(Process Developer / Business Administrator)
Run-time
(End User, Business Administrator, Process Owner)
BPA Suite
BPM Studio
Business Process Composer
BPM Studio
Business Process Composer
BPM Engine
Workspace
Process Portal
BAM

Figure 17: Oracle application development lifecycle (Oracle, 2010)

Modeling; The first phase in the lifecycle is the process modeling phase. In this phase the process analyst captures the real world process and problems in process models. The Oracle BPMS guide (Oracle, 2010) doesn't mention capturing requirements. It suggest that they are already there (on a high level) and that the process analyst captures the detailed requirements as he models the future state processes. The Business Process Analysis (BPA) Suite lets the process analyst create hierarchical process models, from very high level down to a lower level that might be later on implemented as running processes by the developer. The Business Process Composer can be used to collaborate with the process developer and Oracle BPM studio is there to actually design and implement run time processes.

Implementation; This is where the actual system is build, runtime processes are modeled by the process developer (sometimes in collaboration with the process analyst). These tasks also include data mapping, designing user interfaces and creating dashboards.

Deployment; Deployment can be described as the transition phase between development and the runtime environment. The system is integrated with the back-end systems by the business administrator and tested by the developer.

Runtime; At runtime there are four basic tasks that are performed. First, the process participants use the application in Workspace. Second, the process analyst and process owners monitor the real time performance using Workspace dashboards and Business Activity Monitoring (BAM). Third, Process participants who have the necessary permissions can create new processes using Workspace or Process Portal. Fourth, Maintaining the running business applications and the overall run-time infrastructure is the responsibility of the business administrators using the Enterprise Manager.

Advantages and disadvantages of the Oracle BPMS methodology

The advantages and disadvantages of this approach are depicted in Table 10. They are based on the theory described, bearing in mind the goal of creating a tool independent requirements management approach specific for projects using a BPM Suite. In other words, what aspects can be useful and what aspects can become problematic.

Table 10: Advantages and disadvantages of the Oracle BPMS methodology

#	Description	Explanation
Advantages		
A9.1	BPMS specific approach	Oracle BPMS uses an approach specific for BPMS projects (using Oracle). It therefore makes use of the possibilities a BPMS offers.
A9.2	Use of KPIs	The Oracle BPMS methodology encourages to create KPIs.
A9.3	Iteration between phases	The methodology recognizes different phases, but uses an iterative connection between Modeling and Implementation and between Implementation and Deployment.
A9.4	Broad Process Analyst role	The role of the process analyst is broad, which makes the employee flexible and knowledgeable. It includes tasks that belong to a Business Analyst (ROIs), a Business Engineer (KPIs), Requirements Engineer (Requirements) and System Engineer (Process Composer).
Disadvantages		
D9.1	Very tool oriented	The Oracle BPMS methodology focuses to a great extend on the usage of the Oracle BPMS and is therefore not reusable for projects not using Oracle BPM.
D9.2	Very static roles	Roles in the methodology are assigned to Oracle tools, which makes them very static.
D9.3	No explicit description of requirements elicitation	The methodology doesn't describe an approach for capturing of requirements, but assumes they are already there or discovered along the way.

2.4 Summary

The summary of the advantages and disadvantages of the approaches above can be found in Table 11.

Table 11: Summary advantages and disadvantages literature research

Methodology	#	Advantages	#	Disadvantages
Traditional approaches				
RUP	A1.1	Very complete	D1.1	Over complete
	A1.2	Proven solution	D1.2	Prescriptive methodology
	A1.3	Use of UML	D1.3	Used as waterfall
	A1.4	Clear requirements methods	D1.4	Phases are inflexible
			D1.5	No exploitation of BPMS abilities
IRMA	A2.1	Specifically developed for Requirements Management	D2.1	Too many templates
	A2.2	Lots of guidance	D2.2	Waterfall based
	A2.3	Proven solutions	D2.3	Guidelines are highly regulatory
	A2.4	Broad scope in Requirements	D2.4	No exploitation of BPMS abilities
Agile approaches				
Agile RUP	A3.1	Light weight RUP	D3.1	Not that light-weight
	A3.2	More humble	D3.2	Inconsistent definition of Agile RUP
	A3.3	Iterative and incremental within phases		
	A3.4	Hands-on deliverables		
OpenUP	A4.1	Open standard	D4.1	Can be too much information
	A4.2	Lots of Guidance	D4.2	Work products not complete
	A4.3	Iterative and incremental within phases	D4.3	Roles are stubborn
	A4.4	Proven Solution		
Scrum	A5.1	Transparency	D5.1	Full adoption needed
	A5.2	Adaptability	D5.2	Full commitment from client needed

	A5.3	Fast working software	D5.3	Experience needed
	A5.4	Team responsibility		
Visual Requirements Modeling	A6.1	Fast requirements development	D6.1	Not detailed enough
	A6.2	Encourages discussion	D6.2	Business Processes not always visually complex
	A6.3	Helps users know what they want	D6.3	Room for interpretation
Vendor Approaches				
Pega	A7.1	Uses BPMS possibilities	D7.1	Pega specific parts
	A7.2	Just In Time detailed requirements	D7.2	Tooling not flexible
	A7.3	Artifact tooling	D7.3	No requirements data modeling
	A7.4	DCO sessions		
	A7.5	Expectation management		
IBM	A8.1	BPMS specific approach	D8.1	No focus on requirements artifacts
	A8.2	Use of KPIs	D8.2	No iterations from deployment phase
	A8.3	High amount of visual artifacts	D8.3	Requirements stay high-level
			D8.4	No requirements data modeling
Oracle	A9.1	BPMS specific approach	D9.1	Very tool oriented
	A9.2	Use of KPIs	D9.2	Very static roles
	A9.3	Iteration between phases		
	A9.4	Broad Process Analyst role	D9.3	No explicit description of requirements elicitation

2.5 Conclusion

This chapter set out to answer research questions 1:

1. *What requirements management methods and techniques currently exist?*
    a. *In traditional software development?*
    b. *In agile software development?*
    c. *What are the prescribed methods by BPM Suite vendors?*
    d. *Which of these methods and techniques are useful for BPMS projects?*

The chapter has shown the contents of different traditional, agile and BPMS vendor approaches for requirements management. The pros and cons for every approach are stated in Table 11. In general the following conclusions can be drawn:

1. Traditional approaches (RUP and IRMA) are very complete and offer lots of guidance on each subject. The downside is that because of their size, they can be complex, inflexible and prescriptive.
2. Agile approaches (Agile RUP, OpenUP, Scrum and visual requirements modeling techniques) are also complete and give guidance, but thereby assume that adaption is needed later on. The downside is they can still contain an overload of information and require experience.
3. Both approaches do not focus on the business process, neither do they point out the advantages or hurdles of using a BPMS.
4. Vendor approaches (Pega, IBM BPM and Oracle BPM) intrinsically focus on the advantages of a BPMS, but are mostly design and development oriented and therefore do not focus on the requirements process, Pega is the exception.
5. All types of approaches have some elements that have advantages in a BPMS project, they also all have some elements that can cause disadvantages in a BPMS project. A combination is needed.

3 The as-is situation

The previous chapters described the background and state of the art theory regarding requirements management in BPMS projects. This chapter aims to answer research question 2:

2. *How are requirements currently managed in BPMS projects?*
    a. *What are the approaches used in practice?*
    b. *What are the problems encountered?*

To map this current practice, or as-is situation, and to answer these research questions, eighteen interviews have been held amongst practitioners involved with Requirements Management in BPMS projects. The remainder of this chapter describes the reasons for using interviews, the characteristics of these interviews, the analysis of these interviews and the conclusions that answer the research questions.

3.1 Means of research

As said, interviews have been conducted, but what is the reason for doing interviews? Interviews, or qualitative research in general, is intended to explore the world 'out there' (Flick, 2007). To answer these research questions in particular, the world must be analyzed, understood, described and maybe even explained. As (Kvale, 2007) says in his book, it is important to understand the meaning of what is said, as well as of how it is said. An interview gives the opportunity to follow up to certain answers, to explore the depths behind a response. Because every project is different, you as an interviewer, need to be able to zoom in on places where you think lies a best practice, problem or point of improvement. Get the facts behind the opinions and the opinions behind the facts. This is exactly what is needed to answer the research questions above. Quantitative research would be less useful, as our research doesn't only try to find out the as-is situation in a project, but also to find the still unknown improvements, creative ideas that are hidden just below the surface in practitioners minds.

3.2 Characteristics of the interviews

For this interviews a semi-structured interview strategy is used (Alvesson, 2011). It consists of open-ended questions that fall into categories to give some guidelines. In this way, all topics are covered, but in a relatively broad and flexible way, with the ability to ask follow up questions or skip questions if necessary. The questions asked can be found in Appendix A: Interview questions.

The interviews were mostly held with a single interviewee and sometimes with a pair of interviewees. All interviews were held in a face-to-face setting. In total 18 interviews were conducted, these enclosed:

- 5 different industries
    - government
    - banking
    - insurance
    - public utility
    - public transport
- 5 different BPMSs
    - 11 people had Pega experience
    - 3 people had Oracle BPM experience
    - 1 person had IBM Business Process Manager experience
    - 4 people had Cordys experience
    - 1 person had Be Informed experience
    - 2 person didn't have BPMS experience
- Many different roles, but all had experience with requirements

(Alvesson, 2011) argues about the pros and cons of transcripts vs. summary. For this research the choice has been made to use summaries to capture the essence of what is said. This way enough information is captured for analysis in a readable way, whereas transcripts can be enormously time consuming considering not only making, but also reading and sorting the material. To validate the information is captured correct, the resulting summaries were checked with the interviewees and they were able to give feedback or confirm their statements.

3.3 Analysis

The summaries of the interviews can be found in Appendix B: Interview summaries. This section draws up general conclusions from these interviews. Most of the topics covered were explicit interview questions, some topics, such as service integration, came up every time as point for improvement and are therefore mentioned in a separate section.

3.3.1 Environment

BPMS projects take place in a variety of environments. Most projects evolve around connecting back-end systems, automating an existing process or providing a single user interface. All improvements were aimed at efficiency. On average a project takes over 1,5 years. The highest being 5 to 6 years and the lowest being 2 months. For some clients this was their first BPMS project, some had already done a few.

3.3.2 BPMS vendors

In the interviews five different BPMSs were used (Pega, IBM BPM, Oracle BPM, Cordys and Be Informed). Sometimes the project was initiated by Capgemini and then licences were bought for the BPMS and sometimes the project was initiated by a team of the BPMS vendor and Capgemini was brought in for help. A BPMS is mostly focused on the development process, without using an explicit method for requirements, except for Pega which has included their own smartBPM methodology deep into the heart of their suite. You can use the suite without it, but this is harder than when you do use this methodology.

BPMS Vendors mostly posses the skills to show the Proof of Concepts of their products and convince the client of its possibilities, clients are often impressed with the technological possibilities and the speed of development. But not always do vendors posses the methods to turn this technology together with the client into a smooth development process. This is where Capgemini's expertise mostly comes in and this research can contribute.

3.3.3 Project management

Not all interviewees were aware of their project management technique, the answers varied from PRINCE 2, RUP, Waterfall, Scrum, SmartBPM, iterative, etc. Sometimes combinations of these occurred. Mostly when the client used something similar to waterfall and the project something iterative. General consensus among the practitioners is that in a BPMS project an agile project approach (this doesn't mean liberty above all things!) causes less trouble than an non-iterative waterfall approach. As one of the interviewees said: *"The first project was waterfall, so there were nothing but problems in this area. Most of the time it came only to light that this was not how the customer wanted it, in the User Acceptance Test."* Although it is difficult to change the client's way of working, the ones that have succeeded, experienced major improvements in their project. *"First there was a waterfall model for development, this caused that feedback was only given at the end of the development cycle. Things improved when the client started to work iteratively."* An agile process means also a way to match requirements with the agile nature of a BPMS. An example was in a project were they used Scrum: *"Every day the product owner is present at the stand-up. This improves expectation management."*

Next to being agile it is important to get the business involved, create commitment on their side. Multiple projects had troubles, because the client wasn't committed enough and it wasn't until an intervention that the project started to flourish: *"In the beginning there were some problems. Most of the time was consumed by picking up the debris of the earlier months. The second release was much more successful, this was due to: working on client's location, keeping the process tight, making a tight planning and more commitment from the client side."*

In general, an agile and iterative project approach is most suitable in a BPMS project.

3.3.4 Roles

Every interviewee had something to do with requirements, but their roles were not all called requirements specifier, in fact, none of them was. Roles ranged from management roles like engagement leaders to system analyst to solution architect. This tells us something about the nature of the roles in a BPMS environment. Not only their job titles have a broad range, but even more their responsibilities. The requirements job is one that balances between the business field and the IT field. It is suggested to *"use people that understand the business as well as IT. Using a BPMS these roles intertwine and people should focus on hiring employees with these kind of skills."* An example of a skill is to balance use cases, to make them understandable enough and readable enough for the business so they won't just disappear in a drawer and they should be technical enough for the IT developers to work with, without ending up with a lot of question marks.

Requirements elicitation and development are more closely together when using a BPMS, it is important to create direct and iterative communication with the business to align the requirements to the business needs. The new role for requirements is someone who can *"think along with the client"*. Therefore that employee should also have domain knowledge. Take the client through his own process, but bear in mind the technical limitations of a BPMS. When you don't know this information, be able to ask it: *"soft skills of people should be well developed. Using a BPMS intertwines roles in the process. You have to be able to talk to the business as well as develop in the tool. There are lots of people involved and you should be able to communicate with them."*

This new broad role doesn't mean that one man is going to be very busy. It's a 'one for all and all for one' situation. As all roles get broader, everybody gets involved in requirements. As more technical designers should have domain knowledge, functional analyst should have knowledge about BPMSs and processes in general. *"No walls should be build around disciplines, everybody is a specialized generalist."* To conclude, team members in a BPMS project should be broadly skilled. Collaboration is key here.

3.3.5 Elicitation

In most of the projects, requirements are elicited using workshops, interviews and casual conversations with users and subject matter experts. In some cases requirements were already created by the client or some external party, sometimes they were documented in the as is process. In most cases this ready-made requirements ask for more time than they save. *"Asking questions about requirements to the client took more time than it should have. There were no collaborative sessions, if there would have been, this could have decreased development time probably by a month."*

General consensus amongst the interviewees is that the best way to gather requirements is to use collaborative session or workshop, because all relevant skills are present and requirements can be elicited, reviewed and rewritten right away. In the Pega SmartBPM methodology these workshops are called DCO sessions. Detailed requirements are better to gather in individual discussions or documentation, but otherwise the collaborative sessions are very valuable. One of the conditions for the effectiveness of such a session is the presence of people with different skills. *"Most requirements are elicited in the DCO sessions. At this sessions the Business Architect, the systems architects, the Subject matter Experts and the testers are present. Process flows, screens and use cases are used to gather requirements and acquire feedback at the same time. […] The details about requirements and implementation can be done in discussion outside the DCO."*

A collaborative session is highly agile. A business employee can express his wishes, while the system architect expresses the technical implications, which directly results in rethinking the requirement. Prototypes and screens can help to express what is meant to business users.

Regardless of the workshops or collaborative sessions it is important to elicit the requirements iteratively. This means reviews and rewrites. It is impossible to assume that someone can get the requirements spot on the first time. Requirements will change and it's better to anticipate on this. *"Use an agile or scrum like way of working. Use short communication and continuous review of requirements. Keep the business involved to create responsibility."*

In general, elicitation of requirements in a BPMS project works best using iterative workshops, involving all relevant skills and using the BPMS to create speed and consistency in the workshops.

3.3.6 Requirements deliverables

Although the deliverables are vital elements for the new approach, as they are able to give hands-on guidance, in practice there is little consensus about which deliverables to use. Deliverables most mentioned by the interviewees are: Vision document, Business Requirements, Use Case Models, Use Case Specifications, User stories, Business Rules, (high level) process flows, Screen designs, Functional Design (mostly a combination of other deliverables), Story Boards and non-functional requirements/ supplementary specifications.

Less frequently mentioned were: State diagrams, sequence diagrams, interface mappings, Master Correspondence Sheet, chain document, data models, scoping document, Technical Design, Requirements Management Plan, Traceability model, impact analysis,

There was not one project that handled the same set of deliverables, although there was agreement on for instance Use Cases. They also agreed that Use Cases alone are not enough to see the cohesion in a process, a process flow document is also needed: *"a Use Case does not contain the same information as a flow. This would be redundant. A Use Case also explains why a process runs this way. A Use Case and a Flow should in fact act complementary. A flow supports the use cases visually and the use cases give more explanation in the client's language about the flows on parts that cannot be graphically displayed. Together they form the complete picture."*

Several templates were used, most project used the client's templates, a few used their own templates, several used Pega templates and only one project used IRMA templates. Interviewees working with Pega also described the application profile and the application document, which is more or less a generation of other deliverables into one template.

The need for sign offs was very different per project. Sometimes they were very strict: *"Yes sign-offs were needed, everything and by all people involved. This not only reduces risk, but also creates responsibility with the client. It is a good way to define scope."* Other projects were more loose: *"The Application Profile and Application Document need to be signed off, but from the client side as well as from the Pega side there is a lot of trust, and sign offs can be treated flexibly."*

To conclude, a consistent but adaptable set of requirements deliverables is needed, along with guidance on how to effectively use them.

3.3.7 Prioritization

Prioritization is not something that is on average supported by a BPMS. The techniques used amongst the interviewees are for instance MoSCoW (Must, Should, Could and Would), a numbered list or a Scrum product backlog. Most of the time these techniques are only used at a high level, only at the start or not at all. The most common way in BPMS projects to prioritize is in collaboration with the client in an iterative way. *"For prioritization of requirements, it was judged together with the business which requirements had the most business value at that moment."* A simple and flexible way of prioritizing could really be an improvement in BPMS projects. Some exceptions used such a method: "*Requirements are prioritized using the Scrum Backlog. There are multiple Scrum teams, which means multiple Product Owners. Each product owner has its own backlog, but together the product owners have a combined backlog. This way prioritization is well coordinated amongst Scrum teams."*

In general, a simple and flexible prioritization list is suitable for BPMS projects, coping with the agility of a BPMS.

3.3.8 Traceability

Traceability is tricky. It is sometimes supported by the BPMS, for instance in the Pega suite requirements are linked to use cases, to process flows and screens. In general, in the projects of the interviewees, the different requirements deliverables are traceable to one another, but mostly not towards implementation. This mostly has to do with this being a manual step and home-made traceability sheets have to be created. In some projects there is no traceability at all: *"For traceability no actual mapping is made. Most of the traceability is in the heads of the developers."* The reason for this could be that the value of traceability is not always clear, especially in small projects it can cost you more time than it would save. Or because projects work to informal to need traceability: *"Traceability is not really an issue here, because we do not work from signed requirements. If something does not work as expected, we do not go back to read whether it was wrong in the requirements, or if we interpreted them wrong. We just change the system."*

Traceability can become an issue, because in a BPMS environment you anticipate for change. Changes made on implementation level are not always traced back to requirements. Requirements are not always flexible enough, or even better, not agile enough. One of the interviewees wonders: *"As a BPMS is intended to be very flexible, requirements cannot always be changed this suddenly. Should a round-trip be build or should the process be more rigid instead of flexible?"*

One interviewee on a very large project, were traceability was very difficult to implement and maintain, emphasized its importance: *"The advantages of traceability are absolutely clear, it gives speed when handling changes and impact analysis, it also acts as proof when auditing the system and discharging the project team. Traceability is not easy, but it all comes down to discipline. You have to maintain those links."*

In general, traceability is important, also in a BPMS project, but harder to maintain because of the ability to change instantly.

3.3.9 Offshoring

Offshoring is rare in BPMS projects at the moment. Of the eighteen interviews conducted twelve projects were offshoring nothing at all, three projects only did their service integrations offshore, one project used offshored testing, one project used offshored maintenance and one project did actually use people offshore to help with requirements. The latter in this row, that did use offshoring in requirements, wasn't a very big success: *"In the beginning the project was designed to be offshored for 80%, now in reality there are 3 people working offshore. Still the business value of this offshoring is questioned. Because for instance Use Cases need to be translated, offshoring consumes more time than it saves."*

The problem with offshoring in a BPMS project is that, to produce something offshore you need to be very specific about what you want. By the time you explained what you want in enough detail, with the accompanying graphs, you've already partially build your system. Another reason why offshoring is difficult according to the interviewees is that a BPMS operates in a fast changing business environment and to cope with that a close connection to the business is necessary. *"you'll lose your 'one team' spirit and you'll lose the agile edge of the team."*

As can be concluded from the interviewees, it isn't impossible to offshore requirements in a BPMS project, but there is not a lot of experience in practice and a few problems need to be overcome. Extra research is needed to find a solution for these problems.

3.3.10 Testing

When dealing with requirements management, testing is not the first thing that comes to mind, but almost every interviewee mentioned testing as an important party in the requirements process. As said earlier all relevant parties should be included at a workshop or collaborative sessions, and testing should certainly be there. Testers are not that different from requirements specifiers, they both use the perspective: 'how should the system behave'. The only difference is that one comes before development and the other afterwards. As one of the interviewees in a Pega project said: *"involving testers from the requirements stage onwards [...] is a very important best practice to me [...]one of the reasons is that testers have their own very precise way of looking at requirements, which we could notice during DCO session as well."* This own very precise way of looking that testers can have can be because they are trained to look for failures, in other words: how can I try to misinterpret this requirement. A very valuable skill.

A condition for this to work is that testers have enough domain knowledge. Without this domain knowledge, focus is put on less important details, while larger gaps are missed.

3.3.11 Service integration

A BPMS is designed to model processes. The people that work with a BPMS are trained to analyze processes, rethink processes and model these processes. In theory a BPMS is also designed to connect back-end systems as services to the process, but in practice this is a very technical job and involves some challenges. *"Most problems occurred at the services. The Enterprise Service Bus should communicate with the* *legacy systems as well as the three different streams that use Pega. This process caused delays."*

A way to deal with these problems is to start handling service integration very early in the process, capture it in your requirements. This doesn't mean a requirements stating 'we need to be able to connect to service x' is sufficient. Detailed interface mappings and data models have shown useful. But most importantly, one of the interviewees stressed, that these services need to be tested. When still in the requirements face, try to send a simple message to and from a back-end system, try not to trust your first impressions of the interface mapping too much, but evolve this model.

In conclusion, it is important to focus early on service connections. Start this process already in the requirements phase.

3.3.12 Changing requirements

Requirements change, they always do. The handling of change mentioned by the interviewees can roughly be categorized into four groups: No process, a formal process, an informal process and a semi-formal process. First, no change process, one of the interviewees described: *"Requirements changed every week, there was no management for that. No change process caused in this case lots of extra work."* Second, a formal change process: *"There is a large change process with a domain portfolio board who consults about all changes and the impact of those changes."* Third, an informal change process: *"We don't go back to read whether it was stated wrong in the requirements, or if we interpreted them wrong. We just change the system."* And fourth, a semi-formal or iterative change process: *"There was also a change process, but changes were handled mostly in an agile manner. An (informal) impact analysis was made, containing a price tag. When the client still wanted to proceed, you'd know that the change was important enough, but the decision is still the client's in the end."*

The semi-formal or iterative change process is most suited for BPMS projects. BPMS projects often use an iterative way of working. A formal change process would slow down the speed of development. A change process that is to informal or no change process at all, would have negative impact on requirements, people start to lack maintaining them and traceability would fall apart. An semi-formal change process adapted to the iterations of the project would ensure a fairly fast development cycle, while still being able to maintain traceability.

3.3.13 Expectations of new approach

The expectation of the new approach is split up in two. First, the expectations of the content. Second, the expectations of the form. There were a lot of ideas. Regarding the content the interviewees suggested, in summary, the following:

- Create a process that is in gear with the business needs. Therefore include the processes in the requirements, make it Agile and cope with the flexibility of a BPMS. Thereby emphasize the importance of employee soft skills.

- Give guidance on various topics, such as deliverables, traceability and prioritization. Therefore show the best practices and how to make use of the BPMS tools.

Regarding the forms they mostly agreed on three things. First to use the currently available ways to show the depths of the method, adjusted when needed to BPMS use. For example, wikis, (IRMA) templates, guidelines, checklists and Capgemini's Deliver which shows all worldwide methods of Capgemini.

Second, they all said it should also have an overview part. A transparent simple version, which shows the goal of this method, how to use it and what are the elements. Third, it should then have the ability to be tailored to the specific project needs: *"Like a lego box of best practices, but work from the inside out. So start with a small core set of must haves, then create the ability to expand to the needs of the project."*

Some suggested that the product should be sellable to the customer, another idea was that it could even come to certifications for this method. One interviewee brought the out of the box idea to make a movie or hand everybody an iPad with a website, to make it more attractive.

3.4 Conclusions

This chapter set out to answer the research questions 2:

2. *How are requirements currently managed in BPMS projects?*
    a. *What are the approaches used in practice?*
    b. *What are the problems encountered?*

Eighteen interviews have been conducted, which led to the conclusions below. The conclusions have a more practical viewpoint than the ones in section 2.5. They illustrate how theory is put into practice, on what topics best practices have been developed and on what issues there still is room for improvement. Not only have the interviews demonstrated that there is a need for a requirements management approach in BPMS projects, additionally they have shown concrete requirements for the approach that experts believe are important. These are the conclusions drawn:

3.4.1 Best practices used

1. In a BPMS project an agile project approach causes less trouble than an non-iterative waterfall approach
2. Requirements roles in a BPMS project should be broad, so without building walls around disciplines.
3. The best way to gather requirements is to use collaborative sessions or workshops.

4. The most common way in BPMS projects to prioritize is in collaboration with the client in an iterative way, a simple and flexible way of prioritizing could really be an improvement.
5. Testing is an important party in the requirements process.
6. Start handling service integration very early in the process.

3.4.2 Problems/needs encountered

7. It's important to get the business involved and create commitment on their side.
8. There was not one project that handled the same set of deliverables, although there was agreement on a group of them.
9. It's needed to create an explicit link between the business process and the requirements deliverables.
10. Traceability is still often a manual step. A balance is needed between the effort of traceability in respect to project size and the value that is gained by creating changeable requirements.
11. It isn't impossible to offshore requirements in a BPMS project, but there is not a lot of experience in practice and some problems need to be overcome. Extra research is needed to find a solution for these problems.

3.4.3 Expectations/requirements of new approach

12. The new approach should give guidance on various topics, such as deliverables, traceability and prioritization and show best practices.
13. The new approach should be in gear with the business's need for flexibility, agility and employee soft skills.
14. The new approach should make use the currently available approaches to show the depths of the method, adjusted when needed to BPMS use.
15. The new approach should have an overview part. A transparent simple version, which shows the goal of this method, how to use it and what are the elements.
16. The new approach should have the ability to be tailored to specific project needs.

3.5 Relation to conclusions from theory

In section 2.5, five conclusions were stated regarding the state of the art theory. This section restates these conclusions and describes their relation with the as-is situation in practice.

1. Traditional approaches are very complete and offer lots of guidance on each subject. The downside is that because of their size, they can be complex, inflexible and prescriptive.
   This also holds in practice, because of their great size and inflexibility practitioners only tend to use the artifacts from traditional approaches, but not their process methods.

2. Agile approaches are also complete and give guidance, but thereby assume that adaption is needed later on. The downside is they can still contain an overload of information and require experience.
   An agile way of working seems the way to go in practice. Projects have widely adopted an iterative way of working and become flexible. However, because the offer is still large, practitioners all use different approaches and require concrete guidance.
3. Both approaches do not focus on the business process, neither do they point out the advantages or hurdles of using a BPMS.
   This is also true in practice, because both types of approaches do not fit in a BPMS project, practitioners must create their own set of what they think is useful or are obliged to use unnecessary deliverables.
4. Vendor approaches intrinsically focus on the advantages of a BPMS, but are mostly design and development oriented and therefore do not focus on the requirements process, Pega is the exception here.
   Practitioners responsible for requirements agree with this, they endorse the need for guidance in combining BPMS with Requirement Management. Practitioners using Pega already experience some requirements integration in the tool, but underline it's need for further development.
5. All types of approaches have some elements that have advantages in a BPMS project, they also all have some elements that can cause disadvantages in a BPMS project. A combination is needed.
   This is what happens in the as-is situation. Practitioners combine the deliverables and methods they think are useful and support this with the BPMS. A unified approach can prevent reinventing the wheel each project.

4 Proposed approach

The goal of this chapter is to answer research question 3:

3. *What is the recommended tool independent approach for requirements management in BPMS projects?*
    a. *What requirements management principles can be applied?*
    b. *How can they be applied?*

Combining the problems and best practices of the interviews with the existing methods from literature a new approach for requirements management is born. One of the conclusions made in chapter 2 was that *"All types of approaches have some elements that have advantages in a BPMS project, they also all have some elements that can cause disadvantages in a BPMS project. A combination is needed."*. The interviews led to valuable insides on which topics from theory where actually used and which parts weren't used or caused problems. As an example, all interviewees agreed that a workshop was the best way to gather requirements, even if they didn't use the technique at the moment. On the other hand, there wasn't a single project that used the same set of deliverables, even if they used the same approach. Furthermore interviewees stated that the approach should at least comply to the following requirements:

1. The new approach should give guidance on various topics, such as deliverables, traceability and prioritization and show best practices.
2. The new approach should be in gear with the business needs of flexibility, agility and employee soft skills.
3. The new approach should make use the currently available approaches to show the depths of the method, adjusted when needed to BPMS use.
4. The new approach should have an overview part. A transparent simple version, which shows the goal of this method, how to use it and what are the elements.
5. The new approach should have the ability to be tailored to specific project needs.

The remainder of this chapter states the proposed approach, fulfilling the requirements above. The proposed approach is a combination of approaches already existing either in theory or in practice. On important topics in requirements management, it suggests guidelines and best practices. Each sub-topic includes incremental improvements to what already exists and also builds on existing methods and templates, but as a whole it provides a valuable document, giving insight on how to perform requirements management in a BPMS project, independent of a specific suite.

An important note is that this is this version of the approach is obtained by combining the problems encountered and requirements mentioned by practitioners in the interviews and the existing methods from state of the art theory. This version will be up for review by experts. Chapter 6 will address the validity of this approach using expert reviews and chapter 6 will show an improved approach.

THE REMAINDER OF THIS CHAPTER IS CONFIDENTIAL. FOR MORE INFORMATION, PLEASE CONTACT CAPGEMINI.

5 Validation

In the previous chapter an approach for requirements management in BPMS project is proposed. This chapter aims to answer research question 4:

4. *What is the validity of the method?*

The approach proposed from chapter 4 has been checked for validity by a group of experts, using an open interview form. Reviewers received the approach in advance and were able to express their comments in a one-on-one review session. The reviewers were asked to review on how they perceive:

- Perceived ease of use: is the approach free from difficulty or great effort
- Perceived usefulness: is the approach capable of being used advantageously.
- Completeness: does the approach entail all desired topics and does it include enough detail

The first two are derived from the paper of (Davis, 1989). This paper expresses the influence factors for acceptance of information technology. Although this is not an information technology system, we think these measures are still applicable. The third factor is added to make sure no crucial aspects have been skipped in the proposed approach.

There were five reviewers, a BPMS method specialist, two requirements and BPMS practitioners, a BPMS expert group leader and a Requirements Management expert group leader.

5.1 General conclusions

5.1.1 Perceived ease of use

In general reviewers were satisfied with the perceived ease of use of the approach. It is perceived clear and to the point. The separation of the chapters was clear and the high level model provided enough guidance. Reviewers commented that the advice given, was sometimes a bit advertorial or blunt. The reviewers gave advice on which parts they would like to see clarification. Especially on why certain choices have been made or why certain aspects are deemed important.

5.1.2 Perceived usefulness

The reviewers agreed that the approach is perceived as useful. Especially the way the deliverables are presented. The building blocks give guidance in selecting the right deliverables, by stating which ones are recommended and why or when they should be used. There was also consensus that the guidance given in the rest of the approach is a good extraction of best practices and existing methods and is a valuable addition in this field.

Nevertheless were there several tips on how to clarify aspects like prototypes or traceability. The reviewers agreed that Business Rules should be an "absolute necessity", while prototypes should be "highly recommended". Two reviewers expressed that they would like to see the approach to be more concrete, by expressing a clear viewpoint instead of multiple options, or giving examples. In appendix C several detailed recommendations can be found, including if the recommendation is implemented or not.

5.1.3 Completeness

Overall reviewers thought that the proposed approach covered the main topics of requirements management in a BPMS project. Several detailed suggestions have been made like: the addition of a section about change processes, the inclusions of a thought leaders and the mentioning of software for traceability. The detailed recommendations for completeness can be found in appendix C, including if the recommendation is implemented or not.

5.2 Disagreement

Five reviewers means five different opinions. While they agreed on most topics there are two points of disagreement: Smart Use Cases and Offshoring.

5.2.1 Smart Use Cases

Reviewer 1 said: *"Smart Use Cases can be a valuable addition to the list of 'possible extensions'. In the case where Use Cases become too long or complexity is hard to estimate."*, while reviewer 5 said: *"Smart Use Cases are not necessary in a BPMS project. They mostly help translate the business solution to a technical solution. In a BPMS project, the BPMS is used for this."*

Smart Use Cases are common in agile custom software development projects, they are used to cope with complex and long textual use cases, by putting the functional complexity of a use case in a visual model. As reviewer 5 suggests, this visualization of complexity and translation from business solution to a technical solution, is covered by the BPMS, using high level process models, detailed process models and by linking them with the use cases. In conclusion, smart use cases are at this moment not considered a valuable addition to Requirements Management in BPMS projects.

5.2.2 Offshoring

Reviewer 3 said: *"Create a conclusion on offshoring, take a stand."*, while reviewer 4 said: *"Don't give too much advise on offshoring. As it is a whole other field and requires more investigation."* and reviewer 5 said: *"Offshoring is a whole other field of research. This section should make clear that this research obtained these findings and pieces of advice, important enough to share, but separate research is needed to draw conclusions."*

Middle ground can be found here, by following the advice of reviewer 5. The improved approach will clearly state the conclusions based on interviews in current practice, but will at the same time state that more research is needed in this field to find a appropriate solution.

5.3 Changes made

Based on conclusions above and detailed recommendations from Appendix C, several major and minor changes were made to improve the approach in Chapter 4 and create an improved approach in chapter 6. The following changes have been made.

- The introduction now describes in more detail the reasons and goal of this approach and the importance of the three main elements.
- The 'Agile skilled employees' section is now called 'Agile team members' and describes the skills needed for a BPMS requirements specialist in respect to a regular requirements engineer. Furthermore this section now stresses the need for a thought leader.
- More is explained about the Requirements Management Plan and how to create a lean version of this
- A figure representing the aspects of collaboration has been added.
- An addition to customer collaboration stresses the importance of the collaboration with the existing client's processes and methodologies.
- The offshoring section states the conclusions based on interviews in current practice, but at the same time advices that more research is needed in this area to find an appropriate solution.
- A new column 'when use it' is added for the deliverables.
- The deliverable 'Prototype' is now 'highly recommended'.
- The deliverable 'business rules' is now an 'absolute necessity'.
- The deliverable 'backlog' is integrated in the vision document.
- A separate section is included, explaining in more detail the elements of a vision document.
- The section on 'Use your tools' has been removed, as it didn't add any specific value.
- In the 'trace' section, a visual representation of examples of possible traces has been added. Furthermore traceability software is mentioned.
- A section on 'changing requirements' has been added.
- The 'Requirements Management Plan' is changed to a 'Lean Requirements Management Plan' and a section about this deliverable is added.
- A section on BPMS reference frameworks has been added.
- Overall minor changes have been made to improve readability and understandability.

5.4 Conclusion

This chapter set out to answer the following research question:

4. *What is the validity of the method?*

To validate the approach, it was reviewed by five experts. They've declared the method to be a useful tool to manage requirements in BPMS projects. It is easy to use, although some knowledge about the subjects is necessary. In general it is complete, although some additions were mentioned. The next chapter presents an improved version of the approach, based upon the advice of the experts.

6 Improved approach

This chapter presents an improved version of the approach in chapter 4, based on the recommendations mentioned in chapter 5. The management summary of this approach can be found in appendix D. A reference card can be found in appendix E.

6.1 Introduction to approach

Requirements management is the systematic way to gather, process and maintain requirements for a desired system. A Business Process Management Suite (BPMS) is a software tool that can be used to develop software by modeling business processes and use the tool to continuously improve these processes. Working in a BPMS environment is different from working in Custom Software Development (CSD). When using a BPMS, development is faster, more flexible, process focused and because of this, it requires employees to maintain a close connection with the business. This has its impact on Requirements Management.

This approach shows how to improve requirement management in BPMS projects, by explaining how important requirements aspects can be adapted to BPMS project needs. It's based on existing methods from theory and experiences from practitioners in the field. It's organized into three main elements that contain guidelines and best practices (Figure 18). The first two elements are "Be Agile" and Collaborate", they focus respectively on the needed mindset of the employees and the essential collaborative organization. Together they create the process for requirements management. Only when these two are met the project is able to succeed in the next element: "Deliver", which is aimed at creating the products of requirements management.

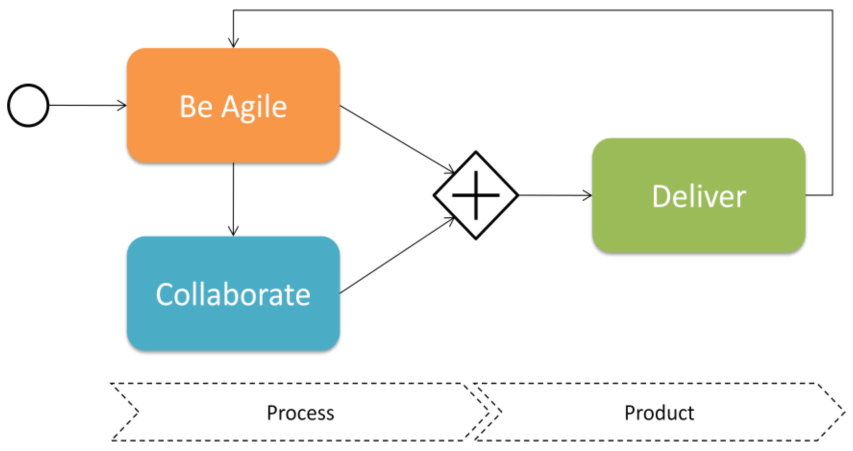

Figure 18: Requirements Management Approach for BPMS Projects

THE REMAINDER OF THIS CHAPTER IS CONFIDENTIAL. FOR MORE INFORMATION, PLEASE CONTACT CAPGEMINI.

7 Conclusions and recommendations

7.1 Conclusions

This research started out stating the following research question:

"How can requirement management in BPMS projects be improved using a tool independent BPMS requirements management approach?"

This question was answered by developing a requirements management approach, specifically aimed at use in BPMS projects. This approach is based on the existing literature and interviews with experts. It was reviewed by experts to improve the approach and confirm its value.
To structure this research it was divided in sets of sub-questions:

- The first set of questions regarded the state of the art in modern theory
- The second set of questions regarded the as-is situation in practice
- The third set was aimed to create a new approach
- The fourth question was to validate this approach

Brief answers to each set of questions will be given in the following paragraphs.

7.1.1 State of the art

This section presents the answer to question set 1:

1. *What requirements management methods and techniques currently exist?*
   a. *In traditional software development?*
   b. *In agile software development?*
   c. *What are the prescribed methods by BPM Suite vendors?*
   d. *Which of these methods and techniques are useful for BPMS projects?*

It became clear that the traditional approaches RUP and IRMA are very complete and offer lots of guidance on each subject, but the downside is that because of their size, they can be complex, inflexible and prescriptive. Agile approaches, like among others Agile RUP and Scrum, are also complete and give guidance, but thereby assume that adaption is needed later on. The downside is they can still contain an overload of information and require experience. Both approaches, traditional as well as agile, do not focus on the business process, the very foundation of a BPMS, neither do they point out the advantages or hurdles of using a BPMS. Vendor approaches intrinsically focus on the advantages of a BPMS, but are often design and development oriented and therefore do not focus on the requirements process, The approach of Pega systems is the exception.

In general, all types of approaches have some elements that have advantages in a BPMS project, they also all have some elements that can cause disadvantages in a BPMS project. A combination is needed.

7.1.2 The as-is situation

This section presents the answer to question set 2:

2. *How are requirements currently managed in BPMS projects?*
    a. *What are the approaches used in practice?*
    b. *What are the problems encountered?*

Interviews were held among experts in the field. Their answers illustrate how theory is put into practice, on what topics best practices have been developed and on what issues there still is room for improvement. Not only have the interviews demonstrated that there is a need for a requirements management approach in BPMS projects, additionally they show concrete requirements, for the approach, that experts believe are important.

Best practices include an agile approach, with broad roles and the use of collaborative requirements elicitation. Furthermore they stress the use of a flexible prioritization process, the inclusion of testers and the early handling of services.

Problems are encountered when collaborating with client's and thereby trying to create commitment, when offshoring parts of the requirements process or when handling traceability. Furthermore interviewees expressed the need for a consistent set of deliverables that also included an explicit link with the business process.

Regarding the requirements management approach for BPMS projects, practitioners require an approach that gives guidance, is in gear with the business's need for agility and makes use of the currently available approaches, adjusted when needed, to BPMS use. Furthermore they expressed their wish to create a transparent simple version of this approach that showed its goals and elements. The approach should also have the ability to be tailored to specific project needs.

7.1.3 The approach

This section presents the answer to question set 3:

3. *What is the recommended tool independent approach for requirements management in BPMS projects?*
    c. *What requirements management principles can be applied?*
    d. *How can they be applied?*

Based on best practices, problems and requirements from the interviews and existing literature, a combined approach has been developed. An approach that shows how to improve requirement management in BPMS projects, by explaining how important requirements aspects can be adapted to BPMS project needs. It's organized into three main elements that contain guidelines and best practices (Figure 19). These three elements are explained in the following sections.

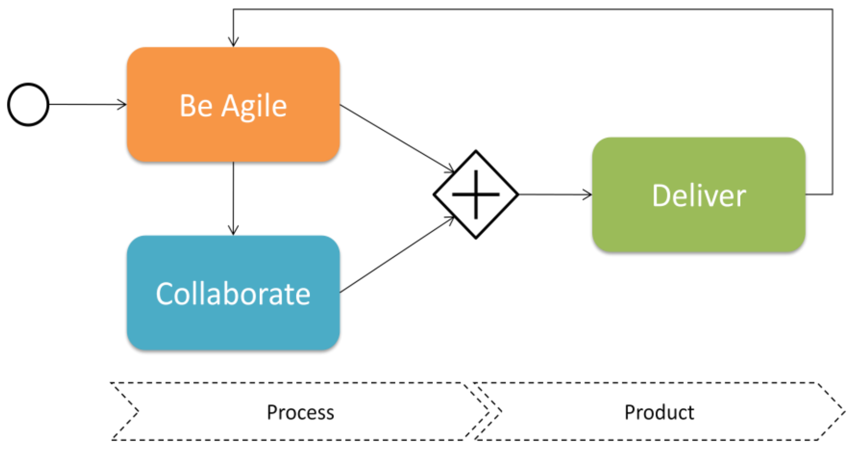

Figure 19: Requirements Management Approach for BPMS Projects

THE REMAINDER OF THIS SECTION IS CONFIDENTIAL. FOR MORE INFORMATION, PLEASE CONTACT CAPGEMINI.

7.1.4 Validity

This section presents the answer to question 4:

4. *What is the validity of the method?*

To validate the approach, it was reviewed by five experts. They've declared the method to be a useful tool to manage requirements in BPMS projects. It is easy to use, although some knowledge about the subjects is necessary. In general it is complete, although some additions were mentioned. An improved version of the approach has been made (and just been described), based upon the advice of the experts.

7.2 Future work

Like every research, this research has its limitations. Future work can help improve the requirements management approach for BPMS projects. Four interesting observations were made throughout this research:

- Practitioners do not yet agree on the use of offshoring in a BPMS project
- The validity of the approach is checked, but not yet guaranteed
- Practitioners didn't mention the use of reference frameworks
- Some deliverable templates are too extensive for BPMS use.

Practitioners do not yet agree on the use of offshoring in a BPMS project. Offshoring is rare in BPMS projects at this moment. This has to do with two problems, regarding specifying in enough detail what you want and creating domain knowledge on the business problem. At this moment offshoring requirements management usually has a negative return on investment. Offshoring requirements in BPMS projects could add value, but to find the right solution further research is needed.

The validity of the approach is checked, but not yet guaranteed. To validate the approach it was reviewed by experts. In general they've declared the method useful and easy to use. However the approach was not tested in practice. Further research could test this approach in one or more a real-life projects. This could further ensure the added value. It's advised to start this case-study validation in a medium sized project (about 10-15 team members and 4-6 months duration). Not too small, so all aspects of the approach can be investigated. Not too large, so the results and consequences can be overseen.

Practitioners didn't mention the use of reference frameworks. Reference frameworks are commonly used frameworks in certain fields. For instance a framework implementing the basics of the process: 'obtaining a mortgage loan'. BPMSs often already posses such frameworks. This might be used as an advantage in the requirements management process, but also means the requirements specialist must know the ins and outs of such frameworks and where these should be adapted. Unfortunately the interviews were not conclusive on whether these reference frameworks create actual return on investment in practice. Further research could depict how these reference frameworks can be used best in requirements management.

Some deliverable templates are too extensive for BPMS use. This research's approach described what deliverables are wise to use, including why and when. For the detailed explanations on these deliverables, the approach references to existing methods. Most of these existing templates are applicable. However some are too extensive for the agile way of working and still need some adaption. Examples are the Vision Document and the Requirements Management Plan, where a first outline has been given. Further research and a real-life test could depict if and how further adaption of deliverables is needed.

7.3 Recommendations

It's recommended to Capgemini employees to use this approach for requirements management in BPMS projects. Applying the approach ensures a better alignment between developing in a BPMS environment and the elicitation, documentation, communication and maintenance of requirements.

It's recommended to use this approach in combination with already existing requirements management methods. Thereby using this approach as a baseline and using the existing methods, on which this approach is build, as background information and references.

It's recommended to store this approach on the knowledge bases Capgemini possesses, to make it accessible for everyone. Thereby including it in the basic training programs to create awareness for the approach.

It's recommended to use the approach when offering for a project, in the sales phase. Thereby demonstrating that Capgemini has a unique requirements management approach for BPMS projects, to overcome problems were past projects might have failed. The approach can be used to create a mutual understanding on the requirements management process and its products, to align with the client's specific challenge.

References

Aalst, W. M., Hofstede, A. H., & Weske, M. (2003). Business Process Management: A Survey. *Lecture Notes in Computer Science , 2678*, 1-12.

Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-Centered Design. In W. S. Bainbridge, *Berkshire Encyclopedia of Human-Computer Interaction* (pp. 763-767). Great Barrington, MA: Berkshire Publishing Group LLC.

Alvesson, M. (2011). *Interpreting Interviews.* London: SAGE Publications.

Balduino, R. (2007, August). Introduction to OpenUP (Open Unified Process). eclipse.org.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). Retrieved February 2, 2011, from Manifesto for Agile Software Development: http://agilemanifesto.org/

BeInformed. (2011). *Business process platform*. Retrieved February 24, 2011, from BeInformed: http://www.beinformed.com/BIWebsite/website/en/EN/ProductSuite

Bergland, J., Maquil, L., Nguyen, K., & Son, C. (2009). *IBM BPM Solution Development Guide.* International Business Machines (IBM) Corporation.

Capgemini. (2011a). *About us: Introduction to Capgemini*. Retrieved February 3, 2011, from Capgemini worldwide: http://www.capgemini.com/

Capgemini. (2011c, March). *Agile RUP*. Retrieved April 1, 2011, from Capgemini Wiki: http://wiki.capgemini.com/index.php/AgileRUP

Capgemini. (2011b). *IRMA*. Retrieved March 3, 2011, from Capgemini Wiki: http://wiki.capgemini.com/index.php/Irma

Cockburn, A. (2001). *Writing Effective Use Cases.* Addison-Wesley.

Davis, F. (1989). Perceived Usefulness, Perceived Ease of Use and User Acceptance of Information Technology. *MIS Quarterly , 13* (3), 319-340.

Divya, V. (2008). Open Unified Process. *Seminar on Open Unified Process* . Kakkanadu, Kerala, India: Cochin University of Science and Technology.

Eriksson, H., Penker, M., Lyons, B., & Fado, D. (2004). Chapter 5 - Dynamic modeling. In H. Eriksson, M. Penker, B. Lyons, & D. Fado, *UML 2 Toolkit.* John Wiley & Sons.

Evans, G. K. (2006). Agile RUP: Taming the Rational Unified Process. In B. Roussev, & L. Liu, *Management of the object-oriented development process* (pp. 231-245). Hershey, PA: Idea Group Publishing.

Flick, U. (2007). *Managing Quality in Qualitative Research.* London: SAGE Publications.

Gartner Inc. (2010). *Magic Quadrant for Business Process Management Suites.*

Gould, J. D., & Lewis, C. (1985). Designing for usability: key principles and what designers think. *Communications of the ACM , 28* (3), 300-311.

Grady, R. (1992). *Practical Software Metrics for Project Management and Process Improvement.* Prentice Hall PTR.

Gregor, S. (2006). The Nature of Theory in Information Systems. *MIS Quarterly , 30* (3), 611-642.

Hirsch, M. (2002). Making RUP Agile. *OOPSLA '02: OOPSLA 2002 Practitioners Reports* (p. 8). Seattle: ACM.

IBM Corporation. (2011). *IBM software - websphere process server*. Retrieved February 24, 2011, from IBM: http://www-01.ibm.com/software/integration/wps/

IBM. (2009, August 31). IBM Business Process Management Prescriptive Guide.

Klabbers, J., Spier, C., Zijlstra, S., & Aalberts, A. (2011). *IRMA wiki portal*. Retrieved March 17, 2011, from Capgemini wiki: http://wiki.capgemini.com/index.php/Irma

Kruchten, P. (2003). *The rational unified process: an introduction.* Addison-Wesley Longman.

Kvale, S. (2007). *Doing Interviews.* London: SAGE Publications.

Leffingwell, D., & Widrig, D. (2000). *Managing software requirements: a unified approach.* Addison-Wesley Longman.

McCoy, D. W., & Cantara, M. (2010). *Hype Cycle for Business Process Management.* Gartner.

Mendix. (2011). *Mendix - Product*. Retrieved February 1, 2011, from Mendix: http://www.mendix.com

Nuseibeh, B., & Easterbrook, S. (2000). Requirements Engineering: A Roadmap. *Proceedings of the Conference on The Future of Software Engineering* (pp. 35-46). Limerick: ACM New York.

Oracle. (2010, July). Oracle Fusion Middleware: Modeling and Implementation Guide for Oracle Business Process Management. Redwood City, CA.

Pega Developer Network. (2010, May 7). *DCO 6.1 - About Direct Capture of Objectives*. Retrieved April 1, 2011, from Pega Developer Network: http://pdn.pega.com/Devnet/PRPCv6/KB/26133.asp

Pega Developer Network. (2011, March). *Implementation & Methodology Overview*. Retrieved April 1, 2011, from Pega Developer Network: http://pdn.pega.com/DevNet/Overviews/MethodologyOverview.asp

Pegasystems Inc. (2011). Retrieved February 1, 2011, from Pega.com: http://www.pega.com

Pichler, M., & Rumetshofer, H. (2006). Business Process-based Requirements Modeling and Management. *First Anual Workshop on Requirements Engineering Visualization (REV'06).* Minneapolis, MN: IEEE Computer Society.

Rational Software. (2003). *Rational Unified Process: Best practices for software development teams.* Cupertino, CA: IBM Corp.

Schwaber, K., & Sutherland, J. (2010, February). *The Scrum Guide.* Retrieved March 8, 2011, from Scrum.org: http://www.scrum.org

Schwalbe, K. (2007). *Information Technology Project Management.* Boston: Cengage Learning.

Spier, C., & Klabbers, J. (2010). *IRMA KM2.0*. Retrieved March 17, 2011, from Capgemini KM 2.0 Knowledge Management: http://km20.capgemini.com/book/164824

The Eclipse Foundation. (2010). *Open UP wiki work products*. Retrieved March 24, 2011, from Open UP wiki: http://epf.eclipse.org/wikis/openup/

Weske, M. (2007). *Business Process Management: Concepts, Languages, Architectures.* Springer-Verlag.

Zielczynski, P. (2008). Chapter 7 - Creating Use Cases. In P. Zielczynski, *Requirements Management Using IBM Rational RequisitePro.* IBM Press.

Appendix A: Interview questions

Context (Made anonymous later)

X1. For which client is the project?
X2. What is the reason for this project? What is the problem statement or client objective?
X3. What is your role in the project?
X4. How much time is anticipated for the BPMS project?
X5. Where does the development take place (Clients' office, Project center, elsewhere)?

BPMS

B1. What BPMS is used?
B2. Why is this BPMS chosen?
B3. How is the project supported by the BPMS supplier?

Roles

R1. Which roles are specified in the project?
R2. Which roles are involved in the Requirements Management part of the project?
R3. Is specific BPMS and requirements training needed for these roles?

Methods and Techniques

M1. What Project Management method is used?
M2. What BPMS development method is used?
M3. What Requirements Management method is used?
M4. How are requirements elicited/analyzed/managed?
M5. How are requirements prioritized?
M6. Are requirements traceable throughout the project?
M7. Are parts of the requirements management process offshored?

Deliverables

D1. What deliverables/documents are created?
D2. What templates are used?
D3. Do all of them need to be signed off?

Client collaboration

C1. How are the clients/stakeholders involved in the project?
C2. On what bases are requirements communicated with the client? (Expectation Management)
C3. How are changing requirements handled?

Lessons Learned

L1. What are things you would recommend to use in future projects?
L2. What are things you would not recommend to use in future projects?
L3. Was there anything you needed in the BPMS project, but wasn't available?
L4. What would you like to recommend to the vendors of the BPMS?
L5. What made the client enthusiastic?
L6. On what elements of the project is the client not convinced?

Expectations of Result

E1. What do you expect of the results of this research, an independent requirements Management approach for BPMS projects?
E2. In what form would you like to see this method (web pages, templates, e-learnings)?

Appendix B: Interview summaries

Interview 1

Industry: Government
Role: System analyst and Functional Designer.
BPMS: none
Duration: ± 3 years

Requirements Management
The project was managed using something similar to PRINCE 2 in combination with RUP and iterative development. To elicit requirement RUP artifacts were used. Including Use case models, use case specifications, state diagrams, screen flows, sequence diagrams and interface mappings. They were elicited using workshops, interviews, prototypes and by reading documentation (especially for other systems/services). For these workshops it was very important to use simulated real life situations with the real end users. Requirements were documented using mostly Enterprise Architect en Microsoft Word. Each element in the high level process flow was matched to a Use Case and sub-flows. The Use Cases were useful, only sub-flows would not have included enough detail. The templates used were developed in-house and all deliverables needed to be signed off by the client.

The prioritization of requirements was done in consolation with the client. Every step in the process was at first a manual step. Steps were isolated and the automated. Eventually, every Use Case was traceable to a piece of code. Nothing was offshored, because of the industry.

Collaboration
The development took place at the client's office, this was a must. Capgemini and the client collaborated on this project, where Capgemini supplied all the supply-team roles, like development, testing, architects and systems analyst and the client fullfilled the demand side roles like users. The three system analysts were in charge of specifying the requirements. One was dedicated to User interaction design. The system analysts had standard RUP, information analysis and Use Case training. The client collaboration was very good. The client's employees had assigned time to collaborate with the project.

In the beginning of the project they tried to come up with all the requirements up front. This wasn't very successful and came with some frustration. After a 'fresh start' was made, communication increased with the client and the client took his time to verify requirements and prototypes. This was successful.

Lessons learned
It is important to focus on system integration. Start with a proof of concept that you can communicate with other systems, before designing the rest of your system. It is recommended to use a more agile approach than a waterfall approach. Also keep your governance tight, communicate and steer clearly.

Expectations of new approach
Make the approach emergent. First deliver a high level view, then detail Just In Time. Make Proof of Concepts where you use role play with the users of the system. Make sure you can review the choice of Value versus Investment every time.

Interview 2

Industry: Banking
Role: First Pega developer, then Functional Designer and eventually Team Lead
BPMS: Pega
Duration: ± 2 years

Requirements Management
The project used a mix of different project management techniques. The pega development team used an iterative approach, while the requirements and test team of the client where more waterfall oriented. Only high level designs of requirements were made. There were no user or system requirements. This caused problems and therefore a functional design needed to be made of the existing systems later on. To elicit requirements Use cases and flows were used. Sometimes requirements just came from e-mails. There was a monthly workshop with the user group. There was a feeling that the requirement specifiers from the client couldn't keep up during these sessions. A data model was build using Pega. It wasn't specified as a requirements artifact, except for the Management Information System. Pega templates were used. Officially the client was supposed to sign off everything. The client doubted too much to actually do so. Nevertheless the project was successfully implemented.

There was no real prioritization in the requirements delivered by the client. Priorities were made by the development team in collaboration with the client. Traceability is build in the Pega software. This wasn't used very often. Documenting is not something a technical developer does by nature. The requirements managers from the client didn't had enough understanding of the BPMS to do so. Nothing was offshored.

Collaboration
The project was a collaboration between Capgemini, Pega and the client. Although development took place at the client, because this was close to the users, Pega had the lead in this project. Together with Capgemini they manned the development team. The client delivered the requirements team and another external party delivered the test team. Most developers had training in Pega, RUP and sometimes Architecture.

There was enough time and labor assigned from the client side, but expertise was missing in their requirements team. In the beginning they where hurrying too much, which gave problems in the testing phase.

Lessons Learned
Use monthly workshops with working demo's. Also keep in close contact with the senior users. Next time the division of tasks among the different teams (from pega, Cap, Client and External parties) could be improved. In the end the client was

impressed by the speed of development and the flexibility in system changes, but the quality of the total solution could be better. Because there was not that much documentation, a lot of discussion rose when Requests for Change were made and the product was already live.

To Pega is recommended that they should put more thought in educating the client about their way of working. Show them what is expected from them to overcome problems later on.

Expectation of new approach
It should be aimed at testing activities and have a link with test management. It should balance between taking fast actions and precise documentation.

Interview 3

Industry: Insurance
Role: System Architect
BPMS: Pega
Duration: ± 2 years

Requirements Management
The project uses the SmartBPM method of Pegasystems. This method also includes a part of requirements management. To elicit requirements this project makes use of Use Cases, Flows, Workshops (DCO sessions) and individual conversations with key users or other relevant people. Feedback is given on requirements using Screens during the application review. Sometimes the team experienced scope creep, for instance, even during the construction phase new requirements were still popping up. About every week an application document was delivered, using the help of the Pega BPMS. An note made is that it is important to balance use cases. Most of the time business people don’t like making use cases or don’t have enough understanding of IT to do so, technical staff doesn’t like documentation by nature and mostly postpones the job. There are templates for Use Cases, but it is important to remember that no template is a holy grail. Templates should be adjusted to the situation, they should be customizable. All deliverables needed to be signed off.

To create prioritization first the happy flows are build. Further prioritization happened in collaboration with the client. Traceability was supported by Pega by linking Use Cases to flows. Some services in this project were offshored using the help of an external party.

Collaboration
The project was a collaboration between Capgemini, the client and Pega, where Pega provided the lead system architects. Two system architects perform the role of requirements specifiers. All employees involved with development had Pega training.

The client performed the role of program manager from a distant location. They are not involved in development. The distance between client and development sometimes causes problems. The client has some back-end system experts that are involved. They got some time assigned from the client to work on this project, but this isn't always enough. To communicate requirements there are lots of DCO sessions and other communication moments. There is an application document. The fast translation of the requirements to screens during the show and tell manages the expectations of the client quite nicely. This is a great advantage of Pega.

Lessons Learned
Use people that understand the business as well as IT. Using a BPMS these roles intertwine and people should focus on hiring employees with these kind of skills. When interviewing a client, write down what they said an let them give feedback. Try to avoid jargon when doing this. Local people that speak the native language and can think along with the client are preferred. Don't allow new requirements to come in for too long.

The client was enthousiastic about the simplicity Pega brought in the organization. Sometimes they went from 25 screens to 1, but the project took longer than expected. There were some problems with communicating with the external services. There should be more clarity about this up front.

Expectations of new approach
Requirements are not a target on their own. It's important to involve the political game. Requirements should also be brought to life instead of plain text and should be understandable for different audiences. The form of this new approach should not be to methodical, but give guidance in the project.

Interview 4

Industry: Insurance
Role: System Architect and Stream Lead
BPMS: Pega
Duration: 6 to 9 months

Requirements Management
On project management level a light version of Prince 2 is used. As a BPMS development method Pega SmartBPM is used as a guideline. Terms and features of the method are used when necessary. To elicit requirements Use Cases, Workshops (DCO sessions), service meetings (outside DCOs), and a data model using the pega framework (extentions were made for back-end communication) were used. It is important to know, that in this project, developers and users worked very closely together. Communication runs directly and therefore efficient. An application profile is made at the start, but not at every iteration. The client did not demand this. Business Objectives were not updated in the BPMS. Next to that a Master Correspondence sheet is created. In the Master Correspondence sheet the content information of for instance letters are saved. This way it can be easily edited by users.

There is a high level program that dictates the order of projects. Within a project it is decided in collaboration with the business which requirements get a higher priority. The as-is situation is taken as a starting point and from there the quick-wins are determined. Use cases are linked to flows using Pega. It is important to remember that a Use Case does not contain the same information as a flow. This would be redundant. A Use Case also explains why a process runs this way. A Use Case and a Flow should in fact act complementary. A flow supports the use cases visually and the use cases give more explanation in the client's language about the flows on parts that cannot be graphically displayed. Together they form the complete picture. A Use Case within Pega is a free format, for this project an own Use Case format was build. Nothing was offshored.

Collaboration
The project was a collaboration between Capgemini, Pega and the client, where Pega had the lead. The business analysts, the users and the system architects from Capgemini were involved with requirements management. Everyone except for users had Pega training.

Clients are very involved in the project. They sit nearby and communicate a lot. The employees of the client have learned to understand flows in Pega and the client wants to train more people. Most client employees are assigned almost full-time to work on this project. Expectation Management was no problem due to this direct communication in a high frequency. Most of the time a first version was build, then it was verified by a user before it was formally written down.

Lessons Learned
Provide lots of communication with the client. Get your information at the source and minimize the number of handovers. Be pro-active regarding requirements, think along with the client. Use smaller groups for the DCO sessions, with only one end-user to gain more efficiency during DCOs. To Pega it is recommended to build something similar to the Master Correspondence sheet used in this project.

The client was impressed by the speed of development and the flexibility of the system. Also the way employees of Capgemini thought along with the users of the system. But the generation of letters had some difficulties. It is important to show these during the DCO sessions.

Expectations of new approach
Every BPMS is different and every BPMS is still in development. Things change and this has its effect on the way of working. This is why it could be a problem to develop a tool independent methodology. Another thing is that soft skills of people should be well developed. Using a BPMS intertwines roles in the process. You have to be able to talk to the business as well as develop in the tool. There are lots of people involved and you should be able to communicate with them.

Interview 5

Industry: Insurance
Role: Lead designer and functional designer
BPMS: Cordys
Duration: 5 to 6 years

Requirements Management

The project management for this project is based on a combination of Scrum and Prince 2. As for the BPMS development method, There is a standard called Cordys@Work, but it is not used. Internally developed standards are used. There is no explicit requirements management method. When eliciting requirements most of the time it starts with a User Story from the Product Backlog. From this a storyboard is made, this is mostly done together with the business employees in some kind of workshop session. These storyboards are then translated into service flows, these are translated in to Functional Designs, these are translated into Technical Designs. From the technical designs implementations are build. Sometimes Use Cases are used but not that often. For all deliverables there are templates. These are developed in-house by the client. Not all of these need to be signed off. Only the chain document needs an official approval. The rest is frequently discussed with users. Due to good change procedures and a flexible product backlogs, the lack of sign-offs is seldom a problem.

Requirements are prioritized using the Scrum Backlog. There are multiple Scrum teams, which means multiple Product Owners. Each product owner has its own backlog, but together the product owners have a combined backlog. This way prioritization is well coordinated amongst Scrum teams. The consequence is that you are flexible in your implementation sequence, but this also means that development teams need to be able to shift quickly. Initially all requirements were traceable from storyboard, to service flow, to Functional Design, to Technical Design. Later on requirements were sometimes directly translated to a functional design, which meant less traceability. This is sometimes difficult for new employees.

Collaboration

The client is the project owner. The project hires internal development teams to do the job. These teams can hire external employees (like Capgemini) when necessary. The product owner is a client employee that creates the high level requirements, the functional designer works out the requirements details. Some employees followed Cordys training, but no specific requirements training was required.

There is lots of communication. There are Scrum sprints of three weeks, which end with a demo for the client. Every day the product owner is present at the stand-up. This improves expectation management.

Lessons Learned

The use of Scrum can be recommended, but initially collaboration between the different product owners did not went very well. Product backlogs were not geared to one another. Another thing was that at the start developers were not physically sitting together, this wasn't good for teamwork and efficiency. To Cordys it is recommended to build connections with high level modeling tools like ARIS.

Expectations of new approach

Eventually it should provide a document that can be used by the development team to implement the system. It should take into account the development of the SOA landscape. It should also have some sort of process architecture (like the Process Architecture Model). Regarding the form, templates are required, so that information can be documented in a uniform manner. Also a wiki would be nice to share experiences with the templates and offer guidelines and ways of working.

Interview 6

Industry: Public Utility
Role: Software Engineer, Team Lead, Requirements Developer
BPMS: Cordys
Duration: 2,5 to 3 years

Requirements Management
In this project the supply side (Capgemini) used Scrum. The demand side (client) first used waterfall, but later on used an iterative approach. Requirements are achieved using a lot of individual discussions. Sometimes also workshops are used, but because the team was very dispersed this was somewhat difficult. Requirements were written down in an Excel document and later translated to Use Cases. A data model was made, but this was not taken into account when eliciting requirements. Furthermore Use Cases, data models, activity diagrams (with extensive descriptions), functional designs and non-functional requirements were delivered. The templates developed by the client were used, these had some issues. The requirements could be written down in a format, but this wouldn't support traceability, maintainability nor coupling to other artifacts. Traceability was a large problem in this project. It was not supported by the requirements deliverables nor by the BPMS. The requirements were prioritized using MoSCoW. This worked very well, especially when the client decided to work iteratively. Nothing was offshored.

Collaboration
Development took place at the client. There was no clear reason for this. The client wanted to build up enough competences to eventually take over the project. There was a Cordys training. There was no specific requirements training. People from the client got training in working iterative.

The client was very involved, especially the demand side. The client was in the lead for the project and the demand team was full-time available for this project. But there was not always enough communication. First there was a waterfall model for development, this caused that feedback was only given at the end of the development cycle. Things improved when the client started to work iteratively.

Requirements changed mostly when something was technically not possible. When change request came from business employees, a change process was started. This meant impact analysis and a cost estimation of the change.

Lessons Learned
Use an iterative approach, this ensures more business involvement, also regarding requirements. The methods used by the client for requirements descriptions were insufficient. Traceability was not supported. Another notion is that it is important to involve the test team in the requirements stage of the project. The Cordys tool lacked

of Requirements Management and it is suggested to build a link to other requirements management tools.

The client was impressed with the speed and the flexibility with which processes could be made and adjusted. This under the condition that the underlying services are well organized. They nevertheless thought it was a pity that Cordys relied this much on Java. Code was needed instead of flexible business rules. This decreased reuse.

Expectations of new approach
Something to maintain traceability. Something to cope with the flexibility of a BPMS. As a BPMS is intended to be very flexible, requirements cannot always be changed suddenly. Should a round-trip be build or should the process be more rigid instead of flexible? Regarding the form of the method, there should be a few simple explanations like: short movies, one-pagers and prezi presentations. After that the details of the method could be explained in a document, wiki or maybe even a forum or help function.

Interview 7

Industry: Insurance
Role: Elicitation of requirements, defining user-interfaces and defining services
BPMS: Pega
Duration: ± 2,5 years

Requirements Management
As Pega is used in this project, also the SmartBPM approach is used. This combination works well. They are trying to find a fit with the waterfall approach used in other projects, like the legacy systems. SmartBPM is also used for Requirements Management. Most requirements are elicited in the DCO sessions. At this sessions the Business Architect, the systems architects, the Subject matter Experts and the testers are present. Process flows, screens and use cases are used to gather requirements and acquire feedback at the same time. Things that are generic should be described generic without going into too much detail. The details about requirements and implementation can be done in discussion outside the DCO. A scoping document, Use cases, requirements, Application document, Application profile and service use cases for data mapping are delivered. These documents do need a sign off, but an e-mail is mostly sufficient. Findings from testing should be qualified and discussed with the operations manager. Testing is aligned with requirements because of the iterative way of working.

Regarding prioritization, there is a scoping document which dictates which part of the project should be done. During the sessions it is established which parts to work on specifically. Sometimes management reports are used to help with prioritizing. Traceability could be improved, not everything was linked in the past. Now Pega is used for that. The development of services is done in India with the help of an external party. This is going well and is still improving. At the start there were some communication problems regarding domain knowledge and mapping of data.

Collaboration
Development takes place at the client's office, this is because the users off the system are located at these sites. The client, Capgemini and Pega work together on this project, Pega has the lead. The business architects and systems architects are full-time available for this project, as well as four key users. These users perform the User Acceptance Tests, give input for requirements and development and also create and give training to their colleagues. When a requirements changes this is channelized. Everything out of scope is transformed into a Request for Change. An impact analyses is done and the result is fitted into a release planning.

Lessons Learned
The things that went well are: The collaboration amongst the different project streams, the transformation from a 100% implementation to a 80% implementation and the evaluation of your roles in the project (who is doing what). Things that could be improved are: In the beginning, the testers were not enough involved. It is also important to review the test specifications and when a flow is changed there is no feedback loop to requirements. This way requirements are not always up-to-date.

The client is impressed with the speed with which applications can be developed, but is less convinced about the services. The Enterprise Service Bus should communicate with the legacy systems as well as the three different streams that use Pega. This process caused delays.

Expectations of new approach
It should support the traceability of requirements. Making requirements more flexible, as flexible as a BPMS. Regarding the form it would be a good idea to create a certification for this, or try to get the material included in existing certifications like BISL.

Interview 8

Industry: Insurance
Role: Systems Architect
BPMS: Pega
Duration: ± 1,5 years

Requirements Management
SmartBPM is used as a project management and requirements management techniques in this project. SmartBPM does not give strict guidelines on how to perform requirements management, it suggests some RUP like artifacts, which were filled in adjusted to the needs of this project. To elicit requirements DCO sessions were used (using for instance prototyping) and smaller discussions outside official DCO sessions. As the project concerned the modeling of an As-Is process, the team could also use documentation from for instance back-end systems. An Application Document, including Use cases and flows and the supplementary requirements are created. No data model is required as a requirements document. For the application document the Pega template is used. For the supplementary requirements a template

was developed in-house. Especially the supplementary requirements sheet needed to be signed off.

Regarding prioritization a separation was made between requirements that were detailed up front and requirements that weren't. User Interfaces were not defined. The test team was also involved for tuning of the test cases to the requirements. For traceability no actual mapping is made. Most of the traceability is in the heads of the developers. Pega does support Use Case linking and detailed requirements are kept in an excel file. Nothing was offshored.

Collaboration
Development took place at the client's office, because this way close collaboration was possible with the system users. People from the client were available for this project and the client was very approachable. There was lots of communication with the client on a very frequent bases. Expectation management was therefore no problem. Because of the very open and honest culture and the iterative way of working, changes were signaled early in the process and never a real problem.

Lessons Learned
Distinguish between requirements that will be specified and requirements that won't be specified. For instance in this project the user interfaces were not specified, but how to create the letters was very explicitly specified. Furthermore, the involvement of the test-team in an early stage of the project is very important. Lots of problems occurred at the service development. A lack of good resources caused delay. Another thing is that business rules could have been more aimed for reuse. On a technical level generic parts could be reused, but on a functional level this could be improved. Regarding the Pega tool, the application document could use some improvement, at this moment it is a bit unreadable, because of the large amount of information it possesses and the lack of difference between business flows and technical flows.

Expectations of new approach
Show the difference between traditional Requirements Management and this new BPMS approach. It should put its emphasis on processes and business rules and how to express that in requirements. Also, requirements elicitation and development are more closely together when using a BPMS, it is important to create direct and iterative communication with the business to gear the requirements to the business needs. Regarding the form, I would like to see some set of techniques instead of a prescriptive method. This way development can be kept agile.

Interview 9

Industry: Public Utility
Role: Architect
BPMS: Oracle BPM
Duration: ± 1,5 years

Requirements Management

As a project management method a derived form of Prince 2 is used, also some RUP elements were included. As a BPMS development method Oracle BPM method was used, the high level process models were build in Enterprise Architect. As was stated in the tender, the project team used IRMA for requirements management. This method is unfortunately not adjusted for use in a BPMS project. It is useful for designing the requirements of the services, but not for the process, this is now done using Enterprise Architect in combination with Microsoft Word. To elicit requirements workshops were used. The users, the business analyst and the requirements specifiers were present at these workshops. The details of these workshops were put into Use Cases. Next to the workshops, individual discussions also served as input for requirements. There was some documentation on legacy systems, but not a lot. Based on IRMA templates they created a vision document, Use cases, non-functionals and user stories (a textual description of the actions of an actor, so the Use Cases including the process), this appealed most to the client. There also were some Use Case realizations, which included more technical specifications. Especially the Use Cases and the Non-functional requirements needed a sign off.

For prioritization a simple order was brought to the requirements. Some requirements were necessary and some were not. Requirements were traceable in the sense that requirements as well as non-functional requirements are linked to user stories and use cases, but not in the sense that they weren't linked to process models. This sometimes caused some delay in finding the right flow for the right requirement. Only testing is offshored. This was not working out in the beginning. The people off shore did not have enough domain knowledge, they lacked of insight sometimes and lots and lots of thing had to be explained. After a lot of interactive sessions this was resolved and now the collaboration is smooth.

Collaboration

Development takes place partly at the Advanced development center of Capgemini and partly at the client. At first the development site was completely remote, but this way the development team is closer to the business users. In the beginning there were some problems. Most of the time was consumed by picking up the debris of the earlier months. The second release was much more successful, this was due to: working on client's location, keeping the process tight, making a tight planning and more commitment from the client side (a business analyst was assigned for 50% of his time). Changes are always send to the business analyst, he not only has the knowledge, but also the mandate to make decisions. This way changes can be handled fast and pragmatic.

Lessons Learned

The way it went in the last release was really successful, there was a tight schedule, where everybody worked hard and there was commitment from all parties. Some communication issues occurred at the start of the project, but they are solved now. Recommended is to be clear on communication and commitment expectations.

Regarding Oracle BPM, there was no requirements management included in the BPMS. This could be done with the help of Business Process Analyzer (BPA), but this wasn't available in the version used. Furthermore, the BPMS is quite technical, business analyst are not able to use it and generic business rules are now written down in Microsoft Word. More integration between Business and IT is suggested for the BPMS. Put back the B in BPM.

The client had high ambitions at first, some of them had to be adjusted with help of Capgemini. Nevertheless was the client pleased with the result. The client doesn't notice a BPMS is used really. They are pleased with the latest results, but this is not granted by them to the BPMS.

Expectations of new approach
Expected is a way to capture the process and the business rules. Use Cases alone do not support this at this moment. Regarding form, a good way would be the same way IRMA is presented, using wikis and the knowledge management system (KM2.0).

Interview 10

Industry: Insurance
Role: Developer and functional designer
BPMS: IBM Business Process Manager
Duration: ± 5 months

Requirements Management
The project was treated like a regular project using Prince 2 as a project management method. There was no BPMS development method used, as the project was a kind of a showcase for the rest of the company, so it was the first time they used a BPMS. Some RUP artifacts were used that were specifically needed for this project. There was no real requirements management method nor were the phases of RUP used. To elicit requirements there were a lot of conversations between the business side, the functional side and the services side. There were some workshops where also the IBM people were present. Technical as well as functional documentation was available for the services. Using the clients templates this project used the following requirement deliverables: Visual models, mapping of services, process descriptions and use cases. Sign offs were absolutely needed, because there were no actual iterations in the process. People were trying to get it right the first time.

For prioritization of requirements, it was judged together with the business which requirements had the most business value at that moment. Requirements were always manually traceable, but not in a software supported way. Nothing was offshored.

Collaboration
Development took place at the client's office, because for the client this was their first BPM project, it was supposed to act as a proof of BPM for the other business lines and show its capabilities. The project was a collaboration between Capgemini, the client and IBM, where Capgemini had the lead. The functional analyst was involved in gathering requirements. It is strongly recommended that they should also have knowledge about BPMN and processes in general. The people from the client got IBM training. Because the project was about integration, the requirements were not that difficult and no extra requirements training was needed. Because the project itself

was mostly about connecting services and wasn't very complex, there were no big problems regarding expectation management and changing requirements.

Lessons Learned

While the use of a BPMS caused fast development and flexibility, the service descriptions were not always ready. This should be done more upfront. Also functional designers should have the knowledge about the product, a BPMS and processes in general.

No BPMS can do everything that is required from Business Requirements, so something had to be done using Java. Also IBM Process Server is a very heavy tool and requires a lot of resources. It is recommended to make the tool less heavy.

The client was impressed with the speed of development and the good integration with the back-end systems, but three points could be improved: First, Java controlled the process at some points, this was not what they expected. Second, there were not a lot of people with knowledge of this tool. Third, the services need to be ready upfront.

Expectations of new approach

It should support fast changes from requirements to development, from the modeler to the development tool. It should educate in how functional people can be educated in the BPMS world. It should say something about the selection of the proper tools for a project. Regarding the form is suggested to use the Deliver application of Capgemini where all global methods are present.

Interview 11

Industry: Banking
Role: Project 1: Systems Architect, project 2: Knowledge Analyst
BPMS: Project 1: Pega, project 2: Be Informed
Duration Project 1: ± 2 years, project 2: ± 6 months

Requirements Management

Project 1 used a RUP like project management style, no BPMS development method was used. To elicit requirements sessions with the business analyst were used, the solution was then thought of by the System Architect and Pega in combination with Capgemini started building the system. The system requirements document was delivered and the new rule set. The other deliverables from SmartBPM, like Application Profile, Application Document, Use Cases and Data models were not used at that moment. The system requirements were tested by user representatives, they gave their approval if satisfied. The project used a MoSCoW prioritization and every rule set was traceable to the requirements. Nothing was offshored.

Project 2 used wasn't really clear on the project management style they used, it was kind of an agile way, which meant things changed a lot. No requirements method was used. To elicit requirements, the Architect and the Project Manager talked with the client almost every week. They made a PowerPoint presentation that included the requirements. These requirements were not negotiated with the developers and tended to change every week. This PowerPoint presentation was the only deliverable. It wasn't signed off, it changed every week and there was no real base line for the project this way. The project used a MoSCoW prioritization. Requirements were not traceable. Nothing was offshored.

Colaboration

For project 1 development took place at the client. They wanted to keep everything in doors, for mostly customer privacy reasons. For this reason it was also not possible to connect to the client's systems from the outside. Pega build the ground works of the project, the architecture, after that Capgemini resumes development and Pega is there for support. The system architects had Pega training, the business analyst didn't. Requirements were regularly level set with the Business Analyst and changes were handled using change requests.

Project 2 was developed at Be Informed's office, this way all knowledge was joint together. The development was a collaboration between Be Informed and Capgemini. There were several training programs per role within the project, these were given by be informed. Requirements changed every week, there was no management for that. No change process caused in this case lots of extra work.

Lessons Learned

Use lots of communication. Focus on the solution and not on the technique, the BPMS is not that important to think of a solution for the problem. Furthermore is it important that the requirements analyst should also be someone with knowledge of IT and BPM systems. Furthermore documentation is needed especially when the project is at a contractual bases. When it is a collaboration with the client, less documentation is needed. It is also important to document requirements for testing purposes.

Regarding the tool in Be Informed it wasn't possible to create to complex processes, also it was not possible to keep track of multiple versions of the application. It is suggested to include more methods in the tool as well.

The client was impressed with the speed and flexibility of development, but with Pega expected more resources to be available and with Be Informed a lot of extra coding needed to be done.

Expectations of new approach

It is expected that the method provides templates (for example Use Case templates) adjusted from IRMA for BPMS projects. Furthermore a separation for different project methods (RUP or Agile) is preferred. Regarding the form it is prefered not to use wikis. Templates should be provided in a simple format that can be used by anyone in any role, for instance Microsoft Word.

Interview 12

Industry: Insurance
Role: Solution Architect
BPMS: Pega
Duration: ± 1 year

Requirements Management
In this project SmartBPM is used for the larger part. The project management framework and the testing framework are not used. Requirements are elicited using the interactive DCO sessions. Everybody is involved in these sessions, even the testers. Question are asked to the users. Flows are build which are connected to Use Cases. Feedback is directly given during these sessions. Furthermore, the business analyst has conducted a list, that shows per product, which steps must be taken in the as-is process. This is used as a guideline when building the new process. All deliverables are build in Pega and can be generated into the Application Profile and Application Document. The Application Profile and Application Document need to be signed off, but from the client side as well as from the Pega side there is a lot of trust, and sign offs can be treated flexibly. Some services are offshored with the help of an external party.

Regarding prioritization, the processes are handled in a specific order. At first the plan was to make an overview of all processes and detail them later on, but plans have been changed to first focus on the most important process. Traceability is not really an issue here, because we do not work from signed requirements. If something does not work as expected, we do not go back to read whether it was wrong in the requirements, or we interpreted them wrong. We just change the system. This requires a tester who knows what to expect, so testers in this project must have a lot of domain and process knowledge. There are some more points to make about testing and requiments: First, testers who have little domain knowledge, and who solely rely on requirements, tend to focus on less-important details, and can miss the larger gaps. Second, involving testers from the requirements stage onwards (like in the DCO sessions) is a very important best practice to me; projects where this was the case have worked very well from my perspective. Third, one of the reasons is that testers have their own very precise way of looking at requirements, which we could notice during DCO session as well. Fourth, requirements gathering and testing use the same perspective: "how does the system behave"; the only difference is that one comes before development, the other afterwards.

Collaboration
Development takes place at the client's office, this way direct contact with users (who have the knowledge about current systems) is possible. The project is a collaboration between the client, Pega and Capgemini. Everybody is involved with requirements. Almost everything involving requirements happens at the DCO, where everybody is present. Except for the users and the tester everybody had Pega training, this involved also SmartBPM training, which includes requirements.

The client is well enough involved. One employee of the client is also a developer for the project. The users of the systems have enough time available to discuss requirements. Requirements are communicated frequently, but because of time issues the product cannot be build 100%. Not all users know this yet. Changes haven't occurred yet. There probably is going to be a change process, but a light weight one. Working in this project with a BPMS already assumes you cannot (and you don't have to) know everything in advance. Working with a BPMS enables quick changes.

Lessons Learned

Focus on flows from the start, do not concentrate on use cases first. By focusing on flows you'll trigger the right discussions and be able to get to the right insights with the client. You'll have to get use to not working with fully elaborated requirements. There is not always a document to fall back to. Another thing is that the domain plays a bigger part when using a BPMS. The role of the domain experts, the developers and testers lay closer together. They should be all educated on that.

Expectations of Result
It is important to use the process as a starting point, then link the Use Cases to this process. Make sure the template for process modeling can be used in multiple tools. Try to show the integration with the tooling, don't see everything as a separate entity, show how you can use it in combination with the tool. Regarding the form, try to show the method top-down, show why this method is build and what is its goal. Then proceed to explaining the steps and providing the guidelines.

Interview 13

Industry: Insurance
Role: Engagement Leader
BPMS: Pega
Duration: ± 3300 hours

Requirements Management
Prince 2 is used on the overall program level, but within the projects SmartBPM is used. Also for requirements management SmartBPM is used, especially this first project with Pega is done the Pega way. Maybe in a later stadium projects will evolve to be more of a client mindset. Requirements are elicited using mostly the DCO sessions. Everyone is involved including the testers (testers are satisfied with the very clear Use Cases). The DCO sessions are led by the developer, which is very pleasant, because business needs can be directly translated to implementation issues, which creates direct input/feedback from and towards the client. Flows are modeled and Use Cases are linked to these flows. With the help of the Pega tool Use Cases, flows, non-functionals, Business Objectives, etc. are created. The application profile and application document should be approved by the client.

There is no upfront prioritization. Mostly it's organized in collaboration with the client by looking at the order in the process and business value. Traceability is supported all the way through. This is supported by the Pega tool. Everything is traceable from Business Objectives to Test Cases. Nothing is offshored yet, the client wants to create a knowledge base first. Problems imagined from working offshore is that you'll lose your "one team" spirit and you'll lose the agile edge of the team.

Collaboration
The project has now started at Pega, but should eventually be developed at the client's office. This should create more coherence and client enablement. The project is a collaboration between the client, Pega and Capgemini, in which Pega has currently the lead. All roles are involved in requirements management. Everyone has to do with

requirements. No walls are build around disciplines and everyone is a specialized generalist. All of the employees have had Pega training, there was no specific requirements training for this project, but some people had architecture training.

The communication with the business runs smoothly. There is enough communication to manage expectations. On operational level there are some problems. Workers from the client don't have enough time to spend on the project, they are not 100% committed and are somewhat resistant, because of their career uncertainty. Their employees are mostly architects and functional designers, roles which don't exist in a Pega environment. In the inception phase a first baseline is established. In the elaboration phase changes can be made to this baseline and a second baseline is established. Every change after the elaboration phase is handled by a separate change process. This change process is needed, among others, because the client is not 100% agile in its way of working.

Lessons Learned
The collaboration in the project between Capgemini, Pega and the client works very well. The client is impressed with the possibilities of the technology and the speed of development, but the governance of the project could be improved, organization wise the client is not prepared for this project yet. It is recommended to the BPMS vendor to make the professional services more professional. Pega is very skilled in selling their product, but not so much in doing projects to implement applications at the client. This is mostly where Capgemini comes in.

Expectations of new approach
A lot. It should be able to cope with fast changes. It should be pragmatic and manageable. It should be transparent. It should be agile. Regarding the form it should be like a lego box of best practices, but work from the inside out. So start with a small core set of must haves, then create the ability to expand to the needs of the project.

Interview 14

Industry: Public Utility
Role: Systems Analyst
BPMS: Oracle BPM
Duration: ± 1,5 years

Requirements Management
For this project IRMA is used as a requirements management method. Although it has been adjusted to the needs of this project. For instance service definitions are constructed on a higher level in the organization at the client's side. The same holds for the vision. These documents should be owned by the client and they should feel responsible. The requirements are elicited using workshops where user stories (scenario describtions) play a major role. This way requirements are elicited inductive instead of deductive, this means you start bottom-up with single actions and start working towards the greater picture instead of the other way around. User stories are transferred to Smart Use Cases, which should always be OTOPOP (one time, one

place, one person). In the end there also is a high level process model, which is linked to the Use Cases and work order statuses. Using the IRMA templates the Use Cases, Supplementary Specifications, Vision document, Requirements management plan, Use Case Overview, process flow, Enterprise Architect model and Enterprise architect traceability model are constructed. All of these are build in Enterprise Architect, only the textual use cases are made in Microsoft Word. The Use Cases need to be signed off definitely, they carry the project. The vision document for instance is created by the client itself. Most of the deliverables are just for notification.

Prioritization is done in collaboration with the client. The releases are prioritized, per release prioritization is done in a pragmatic way, by looking at difficulty and business value. Requirements are somewhat traceable, the flows in the BPMS refers to the Smart Use Cases. In the beginning the project was designed to be offshored for 80%, now in reality there are 3 people working offshore. Still the business value of this offshoring is questioned. Because for instance Use Cases need to be translated, offshoring consumes more time than it saves.

Collaboration
The development takes place for three days at the client and two days at Capgemini. This way it is an optimal mix between working at the client, where collaboration with users is high and working at Capgmini, where the team can work more freely and can collaborate with employees in India. The system analysts discuss the requirements with the Business Analyst. The tester is also present and gives feedback on the Use Cases. It is suggested that it would also be pleasant if a developer was present, because a developer knows the limitations of the BPMS and can give direct feedback on technical issues. Only developers and other technical staff need Oracle BPM training. The system analyst both had training as well as experience in the requirements field, but no specific requirements training was required for this project.

In the beginning collaboration was a disaster, now it works quite well. Client employees are very approachable and available. Communication takes place on an almost daily bases and an official level set takes place every week. The business analyst is dedicated for this job from the client's side. A changing requirements is handled using a Request For Change. The change process starts with an impact analysis. This Impact analysis used to be for free, but because this got out of control the client also has to pay for an impact analysis.

Lessons Learned
The way the project is running at the moment they are very satisfied. Especially about the process, the tight planning, the detail level of the Use Cases (not too high, not too low) and the use of User Stories, which enables you to speak the client's language, because a process model can be to complex or abstract. Furthermore it is suggested to use a Agile/RUP approach, not everything needs to be tied down up front. When collaborating with an offshore test team, it would be nice to have a dedicated controller working in the Netherlands. The client was enthusiastic about the Use Cases, the tight planning, the quality of the work and the speed and flexibility of development.

Expectations of new approach
Currently IRMA is used. IRMA misses the cohesion between Use Cases and the process, this should be added to the method. Make use of User Stories to show concrete examples. It should also give a first move for chain testing. Regarding the form, it should be attractive, like a movie. It should be a stepping stone towards a reference work. It would be nice if this reference work can be supplied via a website, so lots of linkages are possible. A good way to get people interested is by putting it on an iPad.

Interview 15

Industry: Public Utility
Role: Solution Architect
BPMS: Cordys
Duration: 2 to 5 months

Requirements Management
This project used Cordys@work as a BPMS development method, but no specific requirements management method was used. Requirements were elicited using interviews, pilots and demos/workshops. These workshops were let by the architect and were attended by the enterprise architect, the management, the quality and testing team and the users. Using client's and own build templates, the deliverables created were a requirements document, a functional design (which included use cases, diagrams and screens), a high level solution design, a technical design and the solution deliverables (the implementation). The Use Cases were in this case stand-alone descriptions. The details of these Use Cases were elaborated in process flows. Everything was signed off, and by all people involved. This not only reduces risk, but also creates responsibility with the client. It is a good way to define scope.

The architect prioritized the requirements (in a numbered order). After that the prioritization was reviewed by the enterprise architect, the business and the quality team. They all needed to sign off this prioritization. Requirements were traceable to the functional design, but not to the implementation. This is also not supported by the BPMS. Maintenance was offshored in this project. A general remark about offshoring is that it's very hard in BPMS projects, because to offshore something it is necessary to highly specify your needs. When working with a BPMS the development environment is already simple. If you're able to explain what the process should look like, you've already build it partially. This is why it is hard to find business value in offshoring BPMS development. Another reason is that BPMS development environments change heavily and a close connection to the business is necessary.

Collaboration
The development of this project took place at the client's office. The project was a collaboration between the client, Capgemini, another external party and Cordys for support. In the first project it was the client's solution architect that provided the requirements. In the second project requirements management was done by the architect. The architect and all roles later involved in implementation had Cordys

training. No specific requirements training was needed for this project. Requirements were managed based on experience, it also helped to let the requirements be reviewed by multiple people in the project.

The client was involved enough in the project. They were available for requirements analysis and reviewing when needed. The client was involved in every step of the process, the steps were iterative and agile. It is important to keep short communication lines. The focus of the architect was not only to keep the business involved, but the development team as well, to create one coherent team. There was also a change process, but changes were handled mostly in an agile manner. An (informal) impact analysis was made, containing a price tag. When the client still wanted to proceed, you'd know that the change was important enough, but the decision is still the client's in the end.

Lessons Learned

Use an agile or scrum like way of working. Use short communication and continuous review of requirements. Keep the business involved to create responsibility, manage this by maintaining the scope and present costs as an argument. Regarding the tool, documentation is not included in the tool. At this moment you need to create separate documents for that. This could be improved. The client was enthusiastic about the speed and flexibility of implementation.

Expectations of new approach

The business people should be able to look into the BPMS, that is where the added value of a BPMS lies. The usability and flexibility can only be created if the business can understand the (high level) process models. Not if they work in the old way of just looking at additional documents, then nothing has changed. Furthermore it is important to use the BPMS for documentation, do not do this separately, the business should sign off the BPM, not the documentation. When modeling in the BPMS make sure that it doesn't get to technical, the business should be able to understand. Regarding form, I would like to see templates, guidelines, checklists and change management guidelines. There also should be some high level process on how to use this method, with the ability to tailor it to your specific project needs.

Interview 16

Industry: Banking
Role: Project 1A: Team Lead, project 1B: System Architect
BPMS: Pega
Duration: ± 1,5 years

Requirements Management

In the first project a waterfall method was used. In the second project a more agile, scrum like approach was initiated. In the first project, the business analyst and the product manager from the client made the user requirements and business requirements. The systems architect from Capgemini created from those documents the systems requirements. After that an impact analyses and a technical design were made before development started. In the second project requirements came as Request for Change, this meant that a few steps in the requirements management process could be skipped. For instance the elaboration phase was not necessary nor were the DCO sessions. Using the client's templates the deliverables creates were the business requirements, the user requirements, the system requirements, the impact analysis, the technical design and the functional design. The technical design included for instance process flows drawn in Visio. The was kind of redundant, because later on they had to be redrawn in the BPMS. An important suggestion is that you should make use of the BPM tool as much as possible to avoid redundancy.

The Business Requirements and the User requirements had a MoSCoW prioritization. When building the system requirements this prioritization was inherited. The business requirements, the user requirements and the systems requirements were traceable to one another, this was done using a traceability sheet in Microsoft Excel. Implementations could not be traced back to requirements. Nothing was offshored.

Collaboration
Development took place at the client's office, which is in spread out over two locations . The project Manager and the development team are located in Amsterdam, the product manager, users and testers are in Leeuwarden. This distance is sometimes difficult. Communication is mostly done using conference calls. The product manager gathers the users' wishes/experiences and communicates these as requirements to the systems architect. Although Pega initiated the project, it now is a collaboration between the client and Capgemini, Pega now only delivers support.

Asking questions about requirements to the client took more time than it should have. There were no collaborative sessions, if there would have been, this could have decreased development probably time by a month. The first project was waterfall, so there were nothing but problems in this area. Most of the time it came only to light that this was not how the customer wanted it, in the User Acceptance Test. Changes are treated as Requests for Chance (RFCs). The client always said it was a defect instead of an RFC (to cut cost). In reality only 20% were defects and 45% were due to unclear requirements.

Lessons Learned
Let experienced people build a Technical Design, before juniors start building, this saves a lot of questions a opposed to building directly from the system requirements. Do not use the waterfall method. It doesn't work in combination with Pega, instead use an iterative and agile approach. Don't separate groups and roles in the process were everybody has his own island, use blended roles and interact with each other. The client was impressed that the system was build for change, fixes could be live in very little time, but the client expected total development to go faster. Mostly this is due to their own way of working. The client still is rusted into waterfall ideas and doesn't want to make use of the opportunities the tooling offers.

Expectations of new approach
It should not focus on one project situation. It should also be able to handle the gap with the client's way of working we have here. Regarding form, it should be sellable

to the customers of Capgemini. It should have a presentation or offering to the outside world that shows the direct advantages towards management. Eventually there will also be a demand for templates and detailed guidelines.

Interview 17

Industry: Public Transport
Role: Business Analyst
BPMS: Cordys
Duration: 8 months

Requirements Management

This project was set up using an iterative approach. No specific BPMS method or requirements methods were used. To elicit requirements, workshop and individual conversations were used. This way Business requirements, process diagrams, Use Cases, screen definitions and business rules were elicited. No data model was created, which was sometimes difficult regarding the integration with the SAP system, also there was a lack of non-functionals and administration requirements, because of a shortage of time. Use Cases, Process Diagrams, Business Requirements, screen definitions and business rules were created as deliverables. They were all combined in the Functional Design. Homemade templates were used for this. The project was a little chaotic so official sign offs were scarce, unofficial verifications were obtained from the subject matter experts.

Requirements were prioritized using a MoSCoW prioritization, but as the relation with the client was a bit damaged at this point, most wishes were simply obeyed. Requirements were not traceable. This was not necessary because the scope of this project was too small. Nothing was offshored.

Collaboration

Development took place at the client's office, the reason for this is not known. The client supplied the business sponsor, three Software Matter Experts and a SAP team, with whom the new development team needed to collaborate for integration. Capgemini supplied the Project Leader, the Project Management Office, testers, Cordys specialists, a solution architect and a business analyst. The Business Analyst was responsible for the requirements, for that he collaborated with the subject matter experts. The solution architect was sometimes involved for review.

The collaboration with the client didn't go as well as it should have gone in the beginning. It wasn't until the team started to improve the requirements process and started to understand the problems of the client thoroughly, that the whole project improved. Changing requirements were discovered in the review and directly changed, due to the iterative way of working. There was no change process, except for large impact changes.

Lessons Learned

The templates and the method used for the requirements management process worked well. It is important to document requirements properly and use an iterative approach to support changes. One of the things that could have been improved was a separation between business level flows and more technical flows, because of timely issues these two were now documented in one deliverable, which caused confusing on both the business side as well as IT side. Another thing is that there was no attention for Business Activity monitoring and Key Performance Indicators, therefore there was no management information available on the processes to make improvements.

Expectation of Result
It should offer a clear scope on the definition of requirements as the scope of requirements starts to blend. A relation between Use cases and business process specifications. A way to cope with technical requirements and a way to Business Activity Monitoring and KPIs. Regarding the form it should fit in the project of Capgemini's BPMS method that is currently being developed.

Interview 18

Industry: Government
Role: Change coordinator and lead analyst
BPMS: -
Duration: 2,5 years

Requirements Management
was RUP intended, but turned out mostly waterfall. This comes with a lot of extra documentation, but this is absolutely necessary in a project as big as this. To be agile, there should also be people with the mandate to speak on behalf of a group and act quickly, this was difficult to realize in this situation. The (business) requirements were made by the client and then presented. Further analysis of these requirements that was often necessary was done using workshops, interviews etc. The disciplines needed were involved in these discussions. Base on the templates provided by the previous project owner various RUP artifacts were created. Sign-offs of these deliverables are not very common, but I'm not sure. At this moment a review process is operational to get clearance anyway.

Changes were prioritized by the client using three criteria. First, who is noticing this problem (does it impact civilians)? Second, how many civilians are impacted by this? Third, does this solution fit within one of our enhancement programs. The ambition was there initially to make everything traceable with a tool. But the specifications were thousands and thousands of pages and this simply was too much to handle. The advantages of traceability are absolutely clear, it gives speed when handling changes and impact analysis, it also acts as proof when auditing the system and discharging the project team. Traceability is not easy, but it all comes down to discipline. You have to maintain those links.

Collaboration
The project is a real joint effort. A lot of people are working at Capgemini offices, in the past also quite a few were working at the client's offices, especially employees involved in requirements. This collaboration works fine, because everybody is very

involved and flexible. There is a large change process with a domain portfolio board who consults about all changes and the impact of those changes.

Lessons Learned

The collaboration was pretty good, there was great commitment from demand as well as supply side. The administration of documents and controlling versions and changes is under control. Requirements management in general could be improved, there wasn't enough traceability, there was scope creep and some things were just to informal for such a large project. The project control also gave some problems, there was a lack of process monitoring and methodologies were sometimes quitted, without replacing them with something new. The overall quality control is point of improvement.

The client doesn't express much enthusiasm, unless it's about the motivation of the people in the team. I guess that when they attracted a third party they had expected everything to run smoothly.

Expectation of new approach

Where does the requirements specifier go between business analyst and developers, because their roles are closer together in a BPMS project. What artefacts need to be delivered? Furthermore the importance of thinking along with the client should be stressed. To quote Henry Ford: “If I'd listened to my customers, they would have wanted faster horses.” Regarding the form a hyperlink structure like RUP uses is very handy, this way you can see the overview of the method but also explore the depths of it at the same time.

Appendix C: Reviews

Table 12: Review 1

Review 1	
Role reviewer: BPMS method specialist	
Comments on perceived Usefulness	Implemented?
• It's a very useable story, it's very clear and to the point	N.A.
• In the Agile Project Process section, a phase can be seen as a stage with a go/no go point. When phases exists within iterations, this is not possible. Consider some extra explanation.	Yes
• The relation between High Level Process Models and Prototypes needs some extra clarification. Prototypes can be used to elicit process flow from operational personnel.	Yes
• The relation between the 'use case specification' and the 'detailed process flow' needs some extra clarification. Use Case Specification can also describe data centric logic that is not part of the BPM.	Yes
• A domain model can be an addition to a glossary, not necessarily a substitute.	Yes
• Keeping business rules from the process models can also improve adaptability of the solution.	Yes
• Smart Use Cases can be a valuable addition to the list of 'possible extensions'. In the case where Use Cases become too long or complexity is hard to estimate.	No, explained in 5.2.1
Comments on perceived Ease of Use	
• Very readable	N.A.
• It would be more common and understandable to use the term 'workflow' instead of 'detailed process flows'.	No, disagreed by others
• It's assumed that readers know what IRMA is.	N.A.
Comments on Completeness	
• The explanations are to the point and concise. Therefore they do not always entail the complete explanation. But the method does not suggests that it's going to be exhausting on every point, so this doesn't matter.	N.A.

Table 13: Review 2

Review 2	
Role reviewer: requirements and BPMS practitioner	
Comments on perceived Usefulness	Implemented?
• It's a useful document, especially the part stating the deliverables and how to use them.	N.A.
• BPMSs revolve around business rules, it's advised to address them as a key deliverable.	Yes
• Some extra clarification is necessary on end-to-end scenario's.	Yes
• Some extra clarification is needed on prototypes, to explain that these are the generated applications by the BPMS during a workshop.	Yes
Comments on perceived Ease of Use	
• A very readable document. Sometimes it's a bit informal, but this is not distracting.	N.A.
Comments on Completeness	
• It might be a good idea to look at the deliverable "solution outline". Although it is more an architecture oriented deliverable, it could help manage interface descriptions, stakeholders and expectations.	No, architecture oriented
• Maybe there are some UML diagrams that could offer some extra deliverables.	No, UML already included
• When strictly looking at Requirements Management the core set seems valid and complete.	N.A.
• There is software available to handle traceability, it could be a good addition to mention these.	Yes
• BPM is involved in continuous improvement of business processes. Right now there is no document mentioning monitoring and KPIs. Something describing what you should do after everything is implemented. This could be included in the Vision document, but it might be a good addition to include a separate document.	Yes, included in Vision document

Table 14: Review 3

Review 3	
Role reviewer: requirements and BPMS practitioner	
Comments on perceived Usefulness	Implemented?
• The approach often describes that multiple solutions are possible. It would be nice to see examples and take a stand if something is or isn't advised.	Yes
• A Requirements Management Plan is a very large document and most of the time to heavy to include in a BPMS implementation. It can be a good idea to create a lean version of this document and include this as "highly recommended" instead of "absolute necessity".	Yes
• When eliciting requirements not only skills, but also knowledge and power are needed.	Yes
• Create a conclusion on offshoring, take a stand.	Yes
• Mention that the deliverables in the project process figure are not exhausting.	Yes
Comments on perceived Ease of Use	
• It would be easier if more viewpoints or stands were given, the make sure your not still puzzled on what to do next.	Yes
Comments on Completeness	
• Change control would be a valuable addition.	Yes
• A section can be added to "Collaborate", that would describe how to combine this approach with the existing client's processes and methodologies.	Included in section 'Customer Collaboration'
• A problem statement is also part of the vision document.	No, future work
• Reference frameworks are mostly part of a BPMS, these can be used to provide fast implementation, but the requirements engineer must be aware of the functionality and limitations of such frameworks. A mention of this would be valuable.	

Table 15: Review 4

Review 4	
Role reviewer: BPMS expert group leader	
Comments on perceived Usefulness	Implemented?
• The mindset of being Agile is very important and I'm glad it's mentioned first.	N.A.
• It would be a good idea to integrate concrete examples or some advise at the end of a section, this can help, for instance managers, to figure out what actions they must take.	Yes
• Probably in the future the parts about agility and collaboration will get more and more important, while the focus on deliverables will be more integrated in the tool.	N.A.
• Don't give too much advise on offshoring. As it is a whole other field and requires more investigation.	Yes
• The absolute need for a trust relationship with your customer is a very valid point. It's also important to think about the "why"-question, together with your client.	N.A.
• The sentence "Stop designing before the problem is defined" is confusing. It might be a practical problem, but no method suggests this approach.	Yes
• Indeed, it's very good advice to use your tools as much as possible.	N.A.
Comments on perceived Ease of Use	
• Very satisfied with the high-level model of the approach. The separation of "Be Agile", "Collaborate" and "Deliver" and their relations are spot on. It really shows a separation between the person, the company and the artifact.	N.A.
Comments on Completeness	
• It could be a valuable addition to add something about the tasks for each role with respect to the difference in waterfall and agile projects.	No, out of scope
• In addition to "Be Agile", it's also necessary to minimize handovers, to get information at the source and to create a lean process.	Yes, included in 'Agile team members'
• It's important to have a "thought leader" in your team. Someone who can communicate the agile mindset to external parties and stakeholders. This way you can prevent resistance towards working agile.	Yes

Table 16: Review 5

Review 5	
Role reviewer: Requirements Management expert group leader	
Comments on perceived Usefulness	Implemented?
• The first two skills, stated in the agile skilled employees section, are not BPMS specific, but are skills any requirements specifier should have. What this section is trying to say is that the role gets broader and blends more with other roles, therefore you'll have to be a team player. Make that more clear.	Yes
• Explain why there should be a Requirements Management Plan. What is in it? Use it to tailor your project and to deliver the right deliverables from the building blocks.	Yes
• Offshoring is a whole other field of research. This section should make clear that this research obtained these findings and pieces of advice, important enough to share, but separate research is needed to draw conclusions.	Yes
• Explain in more detail, the added value of the building blocks. For instance: packages already on the market have a very broad offer, which can complicate selection of the useful deliverables.	Yes
• Some deliverable templates can also be found in IRMA, not only in RUP.	Yes
• Prototypes are not absolutely necessary, you can run a project without them. It's probably better to include them as highly recommended.	Yes
• Business rules are what drives a BPMS, it should be an absolute necessity.	Yes
• The backlog is always important when working in iterations. It's probably better to include this in the vision document. Thereby noting that this object will change during the project.	Yes
• It would be of great value to add a column "when to use it" to the highly recommended and possible extensions tables. This could include for instance, that you should always use interface mappings, when having five or more interfaces.	Yes
• Smart Use Cases are not necessary in a BPMS project. They mostly help translate the business solution to a technical solution. In a BPMS project, the BPMS is used for this.	Yes
Comments on perceived Ease of Use	
• Nice document, it gives a complete view of the aspects of importance. The deliverable building blocks can be of direct value in Capgemini projects.	N.A.
• It's sometimes written too loose. It shouldn't feel like an advertisement, explain more why certain aspects are important.	Yes
Comments on Completeness	
• The introduction misses some vital information about: what is	Yes

Requirements Management, what is a BPMS, what research has been done, how was this approach established?	
• The Business Analysis Approach of Capgemini called SEMBA, has some interesting views about high level business modeling that might be included.	No, out of scope
• System wide requirements should also state quality attributes.	Yes
• Traceability also gives clarity on what requirements from the vision are realized.	Yes
• A diagram stating an example of traceability could clarify how to use this.	Yes
• Include a section about the change process.	Yes

Appendix D: Management Summary of Approach

THIS CHAPTER IS CONFIDENTIAL. FOR MORE INFORMATION, PLEASE CONTACT CAPGEMINI.

Appendix E: Reference Card

THIS CHAPTER IS CONFIDENTIAL. FOR MORE INFORMATION, PLEASE CONTACT CAPGEMINI.