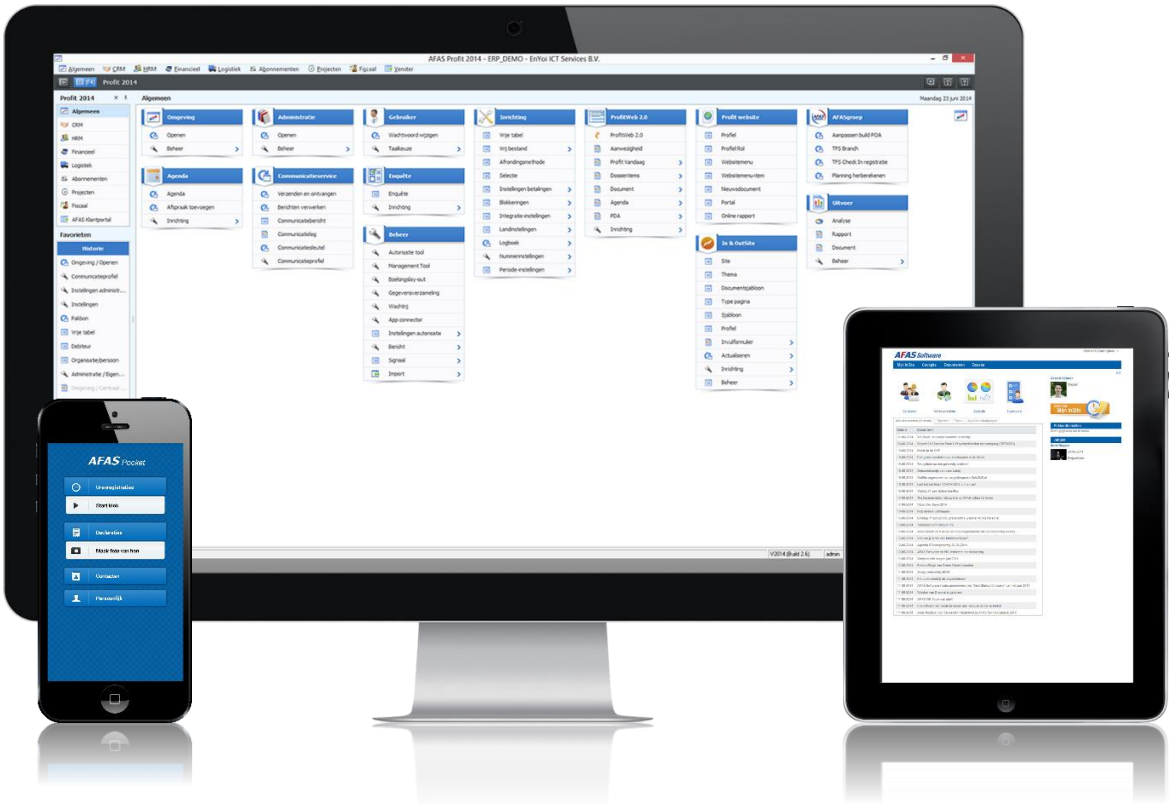
Bridging the Gap Between Software Platforms: A Template Method for Software Evolution

Research thesis

Gerard Nijboer
G.Nijboer@students.uu.nl

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands



Universiteit Utrecht

AFAS Software

Abstract

Software products that no longer meet current customer and market demands are classified as legacy information systems. Legacy information systems can cause problems related to costs, maintenance, accessibility and extensibility. In order to prevent such issues, software evolution by implementing new software platforms in the product portfolio can reveal new opportunities for a software product.

No structured approach towards the mapping of functionality between software platforms exists. A structured approach increases efficiency and allows for comparison and benchmarking of results. The results of an instantiation of the method can be used for product roadmapping, such that a software product manager can make strategic decisions based on the resulting product requirements.

Based on a literature review and expert interviews, a first version of the template method has been designed. By applying the method in a case study, we analyze the template method and its performance. Method increments are designed based on the analysis, which improve the method's next version. The incremental method engineering process is repeated until a stable version of the method is designed. We call this template method the Software Functionality Evolution Method (SFEM).

The SFEM aligns with the Software Product Management Competence Model in three categories, *Triggers*, *Execution* and *Output*. *Triggers* concern focus areas in the Competence Model which initiate an instantiation of the template method. *Execution* designates focus areas that are supported by a template method instantiation. Finally, *Output* focus areas can use the results of an instantiation as input for the execution of their activities.

To allow for reflection on the incremental method engineering process, a categorization for method increments is proposed. This categorization is applied to the method increments and their reasoning and motivation. This concept assists a researcher in explicit reflection of the method engineering process.

Nederlandse samenvatting

Software producten die niet meer voldoen aan de behoeften van klanten en de markt, worden bestempeld als legacy informatiesystemen. Legacy informatiesystemen kunnen problemen veroorzaken omtrent kosten, onderhoud, toegankelijkheid en flexibiliteit. Om dergelijke problemen te voorkomen kan software evolutie nieuwe mogelijkheden blootleggen, door nieuwe software platformen op te nemen in het product portfolio.

Er bestaat geen gestructureerde benadering voor het indelen van functionaliteit tussen software platformen. Een gestructureerde benadering verbetert de efficiëntie en maakt vergelijking en benchmarking van resultaten mogelijk. De resultaten van een instantiëring kunnen worden gebruikt voor het maken van een product roadmap, zodat een software product manager strategische beslissingen kan nemen op basis van de resulterende producteisen.

Op basis van een literatuurstudie en interviews met domeinexperts is een eerste versie van de template methode gemaakt. Door de methode toe te passen in een case study worden de template methode en zijn prestaties geanalyseerd. Method increments worden ontworpen aan de hand van de analyse, welke de volgende versie van de methode verbeteren. Het incrementele method engineering proces wordt herhaald tot een stabiele versie van de methode is ontworpen. We noemen deze template methode de Software Functionality Evolution Method (SFEM).

De SFEM positioneert zich binnen het Software Product Management Competence Model in drie categorieën, *Triggers*, *Execution* en *Output*. *Triggers* zijn aandachtsgebieden in het competentiemodel welke een instantiatie van de methode aanroepen. *Execution* wijst aandachtsgebieden aan die wederzijds worden ondersteund door een instantiëring van de template methode. Als laatste worden de activiteiten binnen de *Output* aandachtsgebieden gevoed door de uitkomst van een instantiëring.

Om de reflectie op het incrementele proces mogelijk te maken, stellen we een categorisatie van method increments voor. Deze categorisatie wordt toegepast op de method increments en hun redenering en motivatie. Dit concept ondersteunt een onderzoeker in het expliciet reflecteren op het method engineering proces.

Preface

This research project would never have been possible,
if it weren't for the people who stood by my side.

Lots of thanks go to my friends, family, colleagues and former colleagues for surrounding me over the past few years. I am thankful for those who surround me, who make me happy, who make me laugh, who help me when I am in need, who are always by my side, simply because they care.

I would like to express my gratitude to my colleagues at AFAS Software, in particular Henk, Mohamed, Bas, Dennis, Daniel, Bart, Mark, Jelle and Floris. Your inspiration, participation, support and knowledge have been a vital contribution to this graduation project and resulting thesis. With joy I look forward to working with you in the near future.

Next, I want to thank my coordinators from the Utrecht University, Jan Martijn and Sjaak, for their contribution to my research. Your dedicated and continuous support, suggestions, advice and reviews have made it possible to get where we currently are.

Also, a word of gratitude goes out to my fellow graduates at AFAS Software. Bas, Jeroen, Rick and Jan Pieter, thanks for your inspiration and contribution during the various moments we have been able to review each other's work.

Most importantly, I want to thank my closest friend, my life companion, my fiancée, Laura. Your everlasting and unconditional love and support have been of great importance to this achievement, which would never have been possible if it weren't for you. Your warmth has given me strength and inspiration to continue, now matter how hard times may have been. With great excitement I look forward to spending the rest of my life with you.

Gerard Nijboer
June, 2014

Communication

G. (Gerard) Nijboer BSc
Graduating student
Student number: 3476766

g.nijboer@students.uu.nl

Utrecht University
Department of Information and Computing Sciences
Buys Ballot Building
Princetonplein 5, De Uithof
3584 CC, Utrecht

dr. ir. J.M.E.M. (Jan Martijn) van der Werf
First supervisor

j.m.e.m.vanderwerf@uu.nl

Utrecht University
Department of Information and Computing Sciences
Buys Ballot Building, office 584
Princetonplein 5, De Uithof
3584 CC, Utrecht

prof. dr. S. (Sjaak) Brinkkemper
Second supervisor

s.brinkkemper@uu.nl

Utrecht University
Department of Information and Computing Sciences
Buys Ballot Building, office 582
Princetonplein 5, De Uithof
3584 CC, Utrecht

dr. H. (Henk) van der Schuur
First external supervisor

h.vdschuur@afas.nl

AFAS Software
Philipsstraat 9
3833 LC, Leusden

M. (Mohamed) Amri
Second external supervisor

m.amri@afas.nl

AFAS Software
Philipsstraat 9
3833 LC, Leusden

Table of contents

Chapter 1

Introduction

This research project covers the subject of software evolution from the perspective of a software product manager. The thesis is subject to the graduation project of Gerard Nijboer for the finalization of his Master's degree in the Business Informatics program at the Department of Information and Computing Sciences, Faculty of Science of the Utrecht University in the Netherlands. Over the period of 2 December 2013 to 30 June 2014 and facilitated by AFAS Software, this research has explored the field of Software Product Management to design a method which assists software developing organizations in the evolution of their software product through mapping of functionality between software platforms.

1.1 Problem statement

As organizations evolve, they embrace new opportunities to strengthen the position of their business in the market. If the products do not evolve as fast the organization, the markets and their mutual demands do, a gap can emerge between the business and its supporting IT systems. In the case where such a software product is no longer suitable for modern needs, nor modifiable for project purposes, it is classified as a *legacy system* (Robertson, 1997). Brodie and Stonebraker (1995) define legacy information systems as "any information system that significantly resists modification and evolution".

Legacy information systems can cause severe problems for organizations, for instance related to costs, maintenance, accessibility and extensibility (Bisbal et al., 1999). In order to correctly cope with these challenges, organizations must find appropriate ways to prepare software products and product portfolios for future needs. Technological advancements can introduce opportunities in order to deal with issues arising from legacy information systems.

The introduction of emerging technologies allows organizations to explore and adopt new opportunities. Emerging technologies can help organizations to innovate, improve efficiencies, and realize new business opportunities (Yee and Oh, 2013). The same goes for software developing organizations, which can use technological advancements to anticipate issues which arise from legacy systems.

For software products, issues can arise from the software platforms it is implemented on. On the other hand, emerging technologies and software platforms may introduce new opportunities. A software platform is defined as "a set of software subsystems and interfaces that form a common structure from which a set of derivative products can be efficiently developed and produced" (Meyer and Lehnerd, 1997). Emerging technologies can introduce new software platforms, such as smartphones, tablets and wearables, which offer new opportunities for software products. The evolution of a software product by implementing its functionality on a new software platform is an example of such an opportunity.

A software product can be deployed onto different platforms, depending on the product's purpose and the requirements of its users. For instance, a product can be deployed on a web-based client for staff operations, a mobile application for existing customers, and a mobile application for en-route sales managers. This results in a variety of software applications on different platforms, each having its own purpose and platform characteristics and constraints.

New technological advancements emerge, develop and mature rather quickly, which makes it a challenge for organizations to make guided, rational decisions on which opportunities to adapt and which to neglect. The decision on which emerging technologies to nominate for implementation is often guided by precision-based methods for technology selection and justification, such as net present value, internal rate of return, payback period and return on investment (Chan et al., 2000). However, as Chan et al. state, information can be unavailable or uncertain, due to a lack of precise and absolute figures which are needed in these calculations. Therefore, decision-makers base their assessment on implicit knowledge, experience and subjective judgments.

When a software product evolves, the software developing organization has to decide which functionality to implement on the new software platform, and which to discard. We call this practice the "mapping" of functionality on software platforms. A mapping, often decided upon by a software product manager, can be based on multiple factors, such as technical and functional characteristics and constraints of the platform, as well as constraints posed by personas.

Currently, no structured approach exists which assists in the evolution of a software product by mapping functionality on new software platforms. This creates a gap in the evolutionary process, as a mapping of functionality between platforms needs to be created, yet it is uncertain what functionality is to be included. The uncertainty of the mapping of functionality can be resolved by exploring what factors constrain a mapping, for instance related to the software platform and the functionality itself. Only if a guideline is set out for the determination of a mapping and its priority, a product management team can follow the same approach in the evolutionary process.

If a software product developing organization is able to correctly apply the proposed method, it can increase agility and efficiency, as the process is guided by a structured approach, rather than pragmatic and subjective reasoning solely based on experience. With more efficiency and transparency in practices, (stakeholder) communication and strategic planning, the organization can respond to changing market requirements more rapidly and thus gain a competitive advantage compared to competitors.

The proposed method assists a software product developing organization in extracting functionality from the software product from a functional perspective, thus without analysis of the software product's technical architecture or source code. This enables a software product manager to compose the requirements for a new application by assigning functionality, without the necessity of having explicit knowledge of the underlying structure of the software product. The focus is therefore purely based on the functional design of software products. The deliverables of an instantiation of the method align with the work of a software product manager, by acting as input for the focus areas *Release definition*, *Roadmap intelligence* and *Product roadmapping* (Bekkers et al., 2010) from the Software Product Management Competence Model (Bekkers et al., 2010). A more elaborate description of how the research fits in the SPM Competence Model is given in Chapter 3.

1.2 Research objective

This research project aims to develop a method which assists in the evolution of a software product by mapping functionality between software platforms. A method is designed as the tool to increase efficiency in Software Product Management practices.

The method is designed to be generic, so that it can be applied to any software product, towards any software platform and thus in any scenario. Especially since the current pace of technological developments is so high, software developing organizations need rapid and efficient contemplation of their options. Each opportunity addressed by the organization requires a quick, yet thorough evaluation of the available functionality. As soon as the mapping of functionality for a user persona on a software platform has been decided upon, the development of the application can begin. And in order to gain the intended competitive advantage, the sooner is the better.

Another objective of the research project is to support the process of strategic roadmapping of a software product. By assigning a priority to the mapping of functionality to a user persona on a software platform, it becomes clear which functionality needs to be implemented at first, and which has a lower priority. Even though an instantiation of the method does not produce a product roadmap itself, the results are perfectly suitable for defining releases, composing a roadmap and identifying strategic themes.

Also related to roadmapping is the objective of efficient stakeholder communication. Software Product Management knows a variety of internal and external stakeholders, such as sales, marketing, development, support, customers and partners. Each stakeholder may need a different view on the plans of the product management department. For instance, a developer may need more in-depth information about the relationships between functionality and the order in which they are to be implemented. On the other hand, a sales manager is more interested in high-level themes of upcoming releases and implemented requirements. Therefore, different and effective visualization techniques for the reporting of results are explored and proposed.

1.3 Research questions

As this research project aims to design a method which assists software product evolution from a functional perspective, the following main research question plays a central role in the research design:

> **What method assists in software product evolution through mapping of functionality between software platforms?**

In order to answer the main research question, the following sub questions address subjects which assist in the engineering of the method:

1. Which methods assist in identification and characterization of software functionality from a functional perspective?
2. What characterizes the users and functional context and constraints of a software platform?
3. Which methods assist in the prioritization of functionality when mapping functionality between software platforms?
4. How can the results of a method instantiation be reported in order to assist in the evolution of a software product?

The validation of the method has not been described as a separate research question. However, it is included by means of the iterative method engineering approach we conduct in the research project. A further elaboration on the research approach is described in Chapter 2.

Sub question 1 aims to identify methods which help an analyst to identify and characterize functionality from a software product. The research uses a composition of method fragments to present a structured approach towards the process of functionality identification. By defining the functionality of the software product in scope, the basis for the proceedings of the proposed method is laid out.

The aim of sub question 2 is to define the target software platform by characterizing the platform's users and functional context and constraints. The resulting parameters give a conclusive definition of the platform, which assists the mapping of functionality to the platform.

When the functionality, users and platforms come together, sub question 3 gives the researcher tools to define the priority of a mapping of functionality. Depending on the demands of the organization, the priority can be determined either by a subjective assessment, as well as by a more structured approach, such as the method by Wiegers (1999a).

Sub question 4 investigates how the results of a method instantiation can be presented to assist a software product manager by providing input for product roadmapping. The visualization of the method's outcome also serves for communicating the results with other stakeholders.

1.4 Relevance

As stressed by Ebert (2007), the success of any product depends on the skills and competences of its product manager. A carefully composed mix of product requirements, tuned to the demands of the variety of stakeholders, is critical to business success. The subject of functionality-centered decision making processes has yet received little attention in accordance with the literature that surrounds software evolution, as is also stressed by Colomo-Palacios et al. (2011).

This research contributes more efficiency in the decision making processes of software product managers, and closes the gap between scientific literature and industrial practices on software evolution. Mentioned earlier in the problem statement in Section 1.1, decision-makers base their assessment on implicit knowledge, experience and subjective judgments, rather than precision-based methods for technology selection and justification.

By deploying the method in the industrial field, further validation and evolution of the method becomes possible. A more advanced and mature method contributes to the Software Product Management Body of Knowledge (ISPMA, 2014), as the practiced decision-making process in evolutionary projects becomes more transparent.

For the industry itself, the method allows for more efficiency in product management practices concerning software evolution, as a structured approach can be adhered to. More efficiency may result in a reduction of costs and consumed resources, and a faster time-to-market. With the opportunities of a more diverse application landscape for a single software product, different market requirements can be met, creating a competitive advantage for the business.

As a more diverse landscape of applications of a software product can be realized, the business faces less threat concerning legacy information systems, reducing overall risk related to costs, maintenance, accessibility and extensibility (Bisbal et al., 1999).

1.5 Main deliverables

The research explores to design a template method which supports a software vendor in the evolution of a software product. A template method is different from a situational method, in the sense that it is not designed specifically for one project. For each project, an instantiation of the template method can be created. This research aims to design a template method, labeled as the Software Functionality Evolution Method.

Other artifacts have been designed during the research project, as well. These artifacts are the instantiations of the template method through case studies, a categorization of method increments, and a matrix to assist in the classification of stakeholders in a method instantiation. Each of the deliverables is briefly explained in the following sections.

1.5.1 Software Functionality Evolution Method

This research project's main deliverable is a template method, known as the Software Functionality Evolution Method (SFEM), which assists a software developing organization in the evolution of a software product by mapping functionality between software platforms. The template method is designed to be generic, which means it can be instantiated in any software evolution project, rather than being specifically tuned to the situational factors in one single project. The SFEM focuses on a functional perspective, to support a software product manager without the necessity of analyzing technical or architectural aspects of the software product.

The method defines a standardized approach towards the extraction of functionality, the definition of user personas and software platforms, and the mapping of functionality based on the applicable constraints and characteristics of those personas and platforms. The template method is presented as a Process-Deliverable Diagram (Van de Weerd and Brinkkemper, 2008) with corresponding activity and concept tables, constructed by method engineering (Brinkkemper, 1996).

Read more about the Software Functionality Evolution Method in Chapter 4.

1.5.2 Template method instantiations

In order to analyze the performance of the template method, the method is instantiated in multiple case studies. These case studies are recorded in a backlog, which describes the process of instantiating the activities and concepts of the method. A summary of these backlogs is given in Section 5.2.

A case study's backlog is an excellent instrument for analysis of the method's performance. The analysis allows for identification of improvements, captured as method increments. As part of the template method's evolution, labels are assigned to the method increments, making reasoning and reflection on the engineering process more transparent and complete.

Read more about the template method instantiations in Chapter 5.

1.5.3 Method increment categorization

To reflect on the incremental process of method engineering, we propose a categorization of method increment types. The application of categories to method increments allows for a transparent reflection on the improvements that have been made, and their underlying reasons. The concept is inspired by the constructivist hermeneutic framework by Van der Schuur (2011), which reflects on the hermeneutic process which is experienced during the construction of artifacts in a research project.

Read more about the method increment categorization in Section 5.1.

1.5.4 Method Stakeholder Classification Matrix

As part of the research, we present the Method Stakeholder Classification Matrix (MSCM). The MSCM proposes a classification for the stakeholders in a template method instantiation, depending on the degree of participation in the template method instantiation and the degree of interaction with the deliverables of the template method instantiation.

Read more about the Method Stakeholder Classification Matrix in Section 4.3.1.

1.6 Case study company

The problem statement is exemplified by a case study at AFAS Software, an Enterprise Resource Planning (ERP) software developing organization from the Netherlands. The AFAS Profit software application currently runs as a Microsoft Windows client application and parts of its functionality is being implemented on a web-based software platform (see Figure 1.1). The web-based platform hosts two applications, AFAS InSite and AFAS OutSite, of which the latter is subdivided in a Portal and a Website (depicted in Figure 1.1 by the horizontal lines connecting AFAS OutSite to Portal and Website).

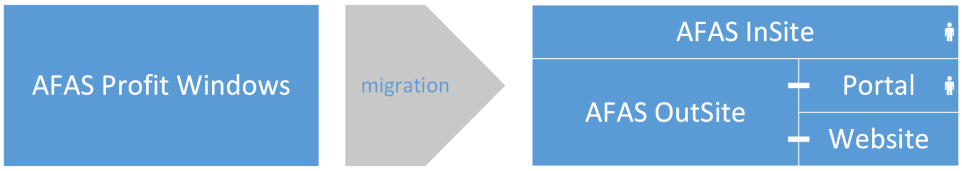


Figure 1.1: AFAS Software platform evolution

AFAS InSite is an intranet application which allows an organization's employees for operations through the web interface, including self-service. AFAS OutSite is an extranet application which enables functionality that should be available to users outside of the organization, such as clients, prospects, applicants and partners. The AFAS OutSite (Portal) requires authentication, just as AFAS InSite (depicted in Figure 1.1 by the user icons on the platforms), and can thus interact with the profile of the authenticated user. The AFAS OutSite (Website) does not require authentication and can be used to present static content about the organization through its website.

At AFAS Software, the Profit Windows application is evolving by extending its functionality over three web-based software platforms, AFAS InSite, AFAS OutSite (Portal) and AFAS OutSite (Website). Each of these platforms hosts one or more different user personas and should therefore host only a subset of the functionality of the full AFAS Profit Windows application. AFAS requires a structured and standardized approach to the evolution, so that the same method can also be applied when for instance a mobile application must be designed for job applicants.

During the evolution of the software product, the organization's product managers have limited resources at their disposal. This forces them to create a software product roadmap containing a rationale on which functionality to implement first, based on the priority given to the different sets of functionality, mapped to the platforms mentioned in Figure 1.1.

The AFAS Profit software product consists of a set of components, each with an extended set of functionality. Extracting functionality and deciding and rationalizing the mapping of the functionality to the different software platforms is a time-consuming task. With the perspective of constantly emerging technologies, for instance related to mobile devices and ubiquitous computing, the organization expects to spend an increasing amount of time on the evolution of its software product, and thus the mapping of its functionality. In order to streamline this process, the organization desires a structured approach, which increases efficiency and ultimately creates a competitive advantage.

The information system at the case study at AFAS Software can only partly be classified as a legacy system, as the AFAS Profit Windows application is still in production and new releases of the application are distributed to the market. The organization still invests a significant proportion of their resources in the development of the Windows product, yet the business is implementing the functionality of the software product on next-generation platforms in order to embrace new opportunities.

1.7 Thesis outline

This thesis is structured as follows: This introduction is followed by an explanation of the research approach in Chapter 2. The research project is contextualized in the field of Requirements Management and Software Product Management in Chapter 3. The Software Functionality Evolution Method and its theoretical foundations are introduced in Chapter 4. The incremental process of template method instantiation is explained in Chapter 5, which shows how the final template method was designed. The thesis concludes with a discussion in Chapter 6, and a conclusion in Chapter 7.

The appendix of the thesis contains a paper submitted to the 8th International Workshop on Software Product Management (IWSPM 2014) in Appendix A, the template method's activity table (Appendix B) and concept table (Appendix C), and two case study instantiations in Appendix D and Appendix E.

1.8 Glossary

The following terms are used in the contents of this thesis. A formal definition is given first, after which the terms will be contextualized in the other chapters of this document.

Classification A grouping of objects on the basis of common characteristics. (IEEE Std. 1671-2010)

Constraint A statement that expresses measurable bounds for element or function of the system. That is, a constraint is a factor that is imposed on the solution by force or compulsion and may limit or modify the design changes. (IEEE Std. 1233-1998)

Data model A data model identifies the entities, domains (attributes), and relationships (associations) with other data and provides the conceptual view of the data and the relationships among data. (IEEE Std. 1320.2-1998)

Design rationale Information capturing the reasoning of the designer that led to the system as designed, including design options, trade-offs considered, decisions made, and the justifications of those decisions. (IEEE Std. 1016-2009)

Entity In computer programming, an entity is any item that can be named or denoted in a program. For example, a data item, program statement, or subprogram. (IEEE Std. 610.12-1990)

Functionality The capabilities of the various computational, user interface, input, output, data management, and other features provided by a product. (IEEE Std. 1362-1998)

Goal An objective that is desirable to meet, but it is not mandatory to meet. (IEEE Std. 1413-2002)

Graphical user interface A means of presenting function to a user through the use of graphics. (IEEE Std. 1387.2-1995)

Mapping An assigned correspondence between two things that is represented as a set of ordered pairs. (IEEE Std. 1320.2-1998)

Method engineering The engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems. (Brinkkemper, 1996)

Method increment A method adaptation, in order to improve the overall performance of a method. (Van de Weerd et al., 2007)

Ontology A description of the entities within the domain in discourse, and how these entities are interrelated. (Gruber, 1993)

Persona Archetypical user of a system, based on research into real users of a system. (ISO/IEC/IEEE 26515)

Product software A packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market. (Xu and Brinkkemper, 2005)

Project plan A document that describes the technical and management approach to be followed for a project. (IEEE Std. 610.12-1990)

Report Information item that describes the results of activities such as investigations, observations, assessments, or tests. (ISO/IEC/IEEE 15289)

Requirement A condition or capability needed by a user to solve a problem or achieve an objective, or a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (IEEE Std. 7-4.3.2-2010)

Roadmap A document that provides a layout of the product releases to come over a time frame of three to five years. It is written in terms of expectations, plans and themes and core assets of the product. (Regnell and Brinkkemper, 2005; Moon et al., 2005)

Scenario The combination of a possible and relevant appearance of a persona on a given software platform.

Situational method An information systems development method tuned to the situation of the project at hand. (Harmsen et al., 1994)

Software evolution The adaption of capabilities and functionality of a system, in order to meet user needs. (Rajlich and Bennett, 2000)

Software migration Transformation of software systems into a new environment, without changing its functionality. (Gimnich and Winter, 2005)

Software platform A platform is the combination of an operating system and hardware that makes up the operating environment in which a program runs (ISO/IEC 26513). Thus, a software platform defines the environment in which a software product is designed to operate.

Software Product Management The discipline and role, which governs a product (or solution or service) from its inception to the market/customer delivery in order to generate biggest possible value to the business. (Ebert, 2007)

Stakeholder Individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations. (IEEE Std. 12207-2008)

Template method A template method describes *what*, rather than *how*, the activities and concepts are to be implemented by the instantiating organization. (Van der Schuur et al., 2011)

User manual A document that presents the information necessary to employ a system or component to obtain desired results. Typically described are system or component capabilities, limitations, options, permitted inputs, expected outputs, possible error messages, and special instructions. (IEEE Std. 610.12-1990)

User need A user requirement for a system that a user believes would solve a problem experienced by the user. (IEEE Std. 1362-1998)

Chapter 2

Research approach

In order to conduct this research project and answer the research questions as in Section 1.3, this chapter continues by describing the applied research approach. Figure 2.1 gives a graphical representation of the research approach, and how this has contributed to the objectives of the research.

2.1 Literature review

To answer the research questions and to provide a solution to the problem statement, the research project initiates by exploring the current state of literature through means of a literature review. The literature review helps to identify and organize activities and concepts which support instantiating the method.

The literature review uses multiple queries with sets of keywords and operators to search online databases of literature for relevant items. Each query focuses on a certain topic, relating to the sub questions, such as “software functionality characterization” or “user persona”. Queries are expanded with operators to include related keywords and synonyms.

Inspired by the PRISMA 2009 checklist (Moher et al., 2009), a structured flow is followed to filter out articles that are not relevant to the research. Figure 2.2 presents the original flow of the PRISMA Statement. At first, all identified articles are gathered and listed together. Then, duplicates from the various queries are removed from the listed articles. The screening phase includes excluding articles based on the contents of their abstract. This gives a high-level overview of the contents of the article, which can be used to assess the relevance of the article to the research. The articles that pass the screening phase, are included in the full-text assessment. This means analyzing the contents of the full-text of the article and extracting contents that are relevant to the research.

The literature review in this research project differs from the original PRISMA Statement by not performing a meta-analysis of the literature. Such quantitative research is relevant to conducting a systematic literature review with the goal of researching the current state of literature on a certain topic, and identifying possible research gaps for further research. However, the current gap in scientific literature has already been identified, and thus a quantitative meta-analysis of the literature would produce too much overhead. This does not serve the goal of the literature review, as the aim is to gather literature to support the template method’s theoretical foundations.

To support the findings of the literature review, and relate them to actual practices in Software Product Management, expert interviews are conducted. These interviews merely serve to validate the outcome of the literature review, and explore new opportunities by means of discussion.

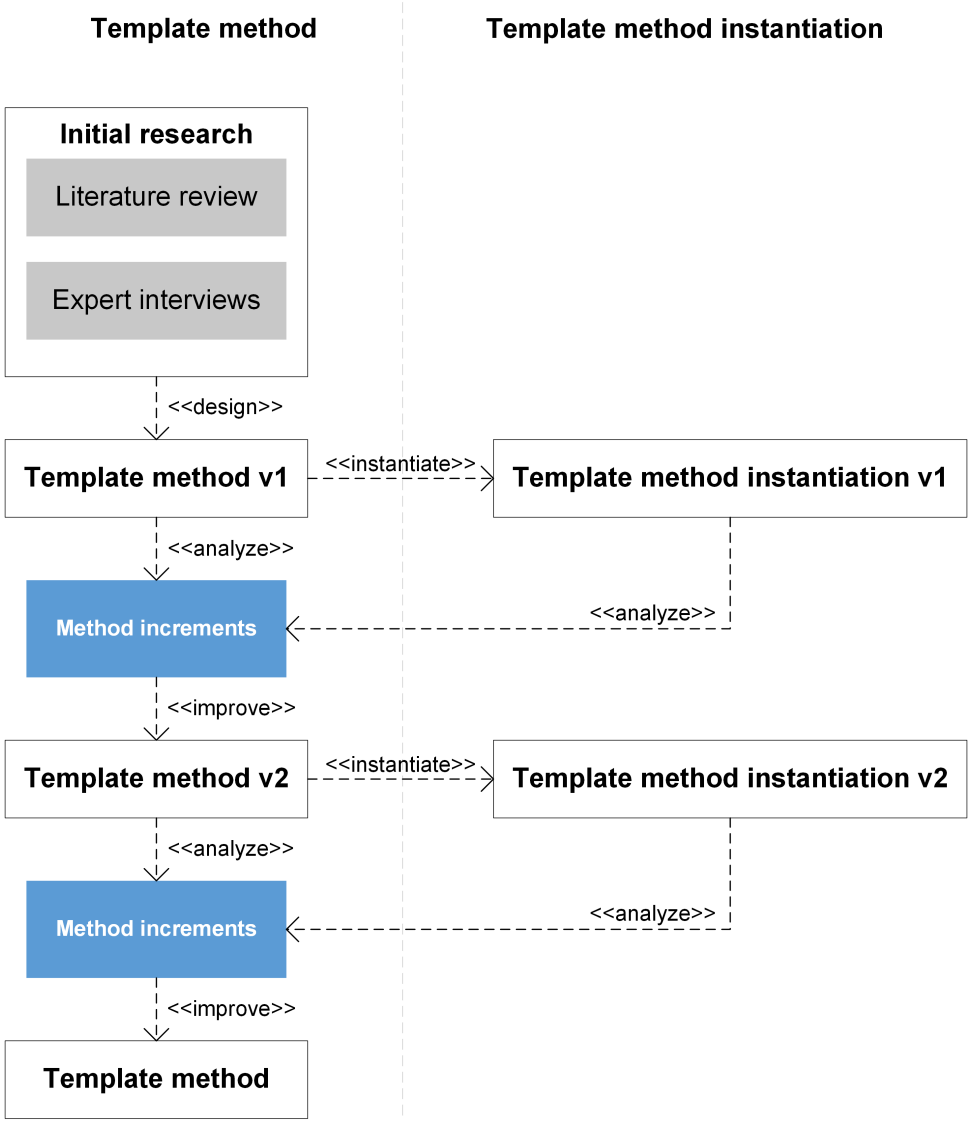


Figure 2.1: Research approach

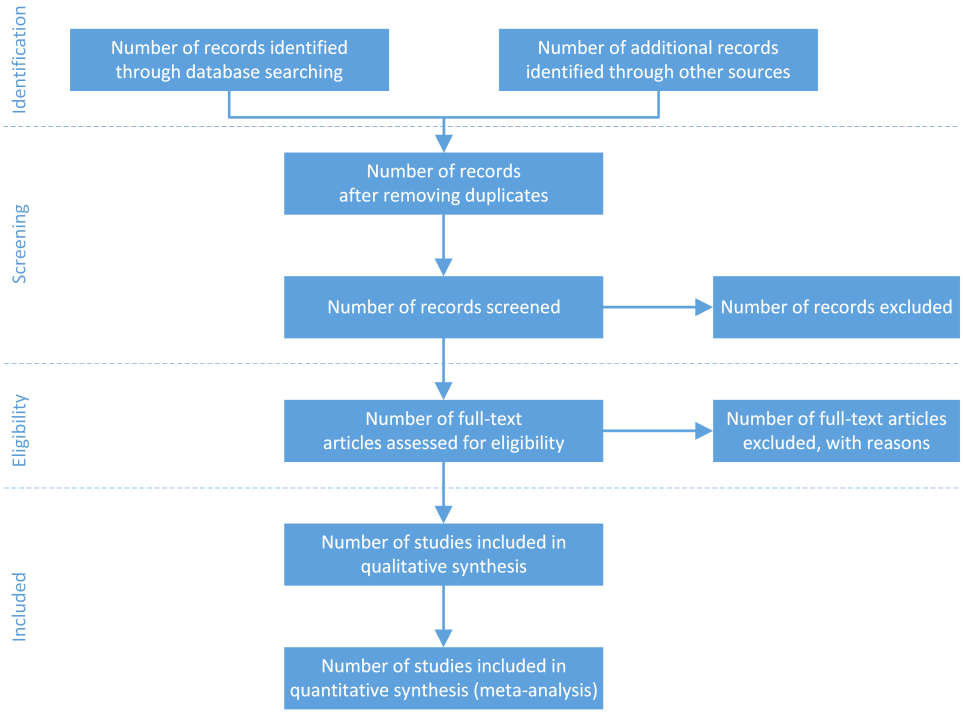


Figure 2.2: PRISMA 2009 Flow Diagram (Moher et al., 2009)

2.2 Method engineering

The current variety of methods for the design, development and implementation of information systems makes it hard to select the correct method for a software evolution situation at hand. Since no method exists which is suitable for every situation, a method can be engineered into a situational method, which is defined as “an information systems development method tuned to the situation of the project at hand” (Harmsen et al., 1994).

The engineering of a method which addresses the problem statement in Section 1.1 and the research questions in Section 1.3 involves the configuration process as described by Brinkkemper (1996) and represented in Figure 2.3. Through the methods administration, method fragments are fed to the selection and assembly of method fragments into the method. The resulting method is instantiated in a project, from which the performance is measured and analyzed.

From the analysis of the method’s performance, method improvements are captured and sent back as requests for adaptations, which results in method increments. The new conceptual method is then again instantiated in another case study, from which the project performance is again analyzed. Only when the project performance results in satisfying output, is the method considered valid and integer.

In Figure 2.3, it is visualized how a situational method is dependent on the situation of the project at hand. The project environment in which the method is to be operable, serves certain project factors, which characterize the requirements for the project. This determines which method fragments are suitable for the method and which are to be omitted.

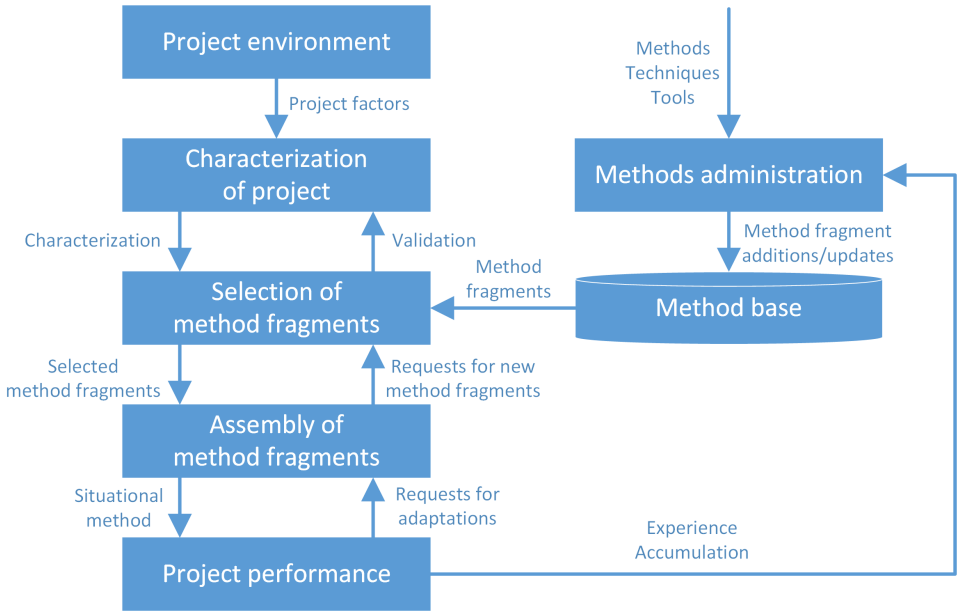


Figure 2.3: The configuration process for situational methods (Brinkkemper, 1996)

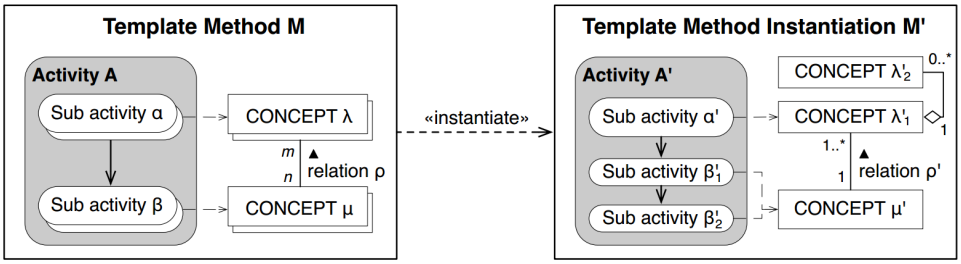


Figure 2.4: Template method instantiation (Van der Schuur et al., 2011)

2.2.1 Template method instantiation

The deliverable of the method engineering process of this research project is a template method, which can be instantiated by software developing organizations to support the evolution of their software product by mapping functionality between software platforms. In contrast to a situational method, a template method describes *what*, rather than *how*, the activities and concepts are to be implemented by the instantiating organization (Van der Schuur et al., 2011). Thus, a template method is different from a situational method as it serves as a template for an instantiation, rather than describing the instantiation of the situational method itself.

In Figure 2.4, the concept of instantiating a template method is visualized (Van der Schuur et al., 2011). The figure indicates how the open activities and concepts of a template method may result in extra elements after instantiation. This can be due to situational factors of the project at hand, which suits the characterization of the project, as depicted in Figure 2.3.

Different from the design of template methods as described by Van der Schuur et al. (2011), we do not design a template method which is only composed of open activities and concepts. This is because many of the described activities and concepts in the Process-Deliverable Diagram are more complex than what is actually described in the method and corresponding activity and concept tables. On the other hand, the method contains simple template activities and concepts which do not require further elaboration before they can be instantiated.

The concept of template method instantiation assists in the method engineering process of this research project. By instantiating the template method in case studies, we are able to analyze the method's performance and identify issues, which can be improved by method increments. These method increments are categorized, so that we can analyze the main improvements made to the method. Chapter 5 gives more insight in how the template method was instantiated in the research project, and what results were gathered from this.

The method instantiation process takes place by initiating a case study, of which the procedure is described in Section 2.3. The method is analyzed in both its template state, as its instantiated state in the case study. This brings forth method increments, which are used as input for the next iteration of the template method.

If the designed template method and its instantiated method performance do not introduce any significant issues, and the results of the instantiation are satisfying, the latest iteration delivers the final template method, which is the Software Functionality Evolution Method (SFEM). This incremental process of composing, instantiating and improving the template method is depicted in Figure 2.1.

To reflect on the process of the research project, we use the constructivist hermeneutic framework by Van der Schuur (2011), which is an adaption to the framework by Cole and Avison (2007). The framework enables us to reflect on the reason why certain increments were made, the cause that lead to the increment, and the classification of the increment.

2.3 Case study

For this research, a qualitative research is conducted through means of a single case study at a product software company. This case study takes place at AFAS Software[1], a Dutch product software company from Leusden which develops an Enterprise Resource Planning (ERP) system called AFAS Profit. The case company serves for inspiration, instantiation and validation of the template method to be developed by means of this research.

In the research, the functionality of the software product is used to map against a next-generation platform, which is a web-based application. By means of this process, which is repeated multiple times on different function groups of the product in order to cover all the functionality, the aim of this research is reached, as the repetitive process of performing the case study eventuallys result in validation of the answers to the sub questions and confirmation that the process in the method indeed helps to identify the necessary concepts.

For the case study, the guidelines by Runeson and Höst (2009) and Yin (2009) assist in structuring the case study. The case study approach matches the objective of this research project, given the characteristics of the research methods named by Runeson and Höst. The primary objective is exploratory, as we are certain there are fragments of the main deliverable available in literature, though the complete method has not yet been composed. This makes us aim at exploring the available data in literature, composing a method, validating the method in a case study, and identifying weaknesses in the research for future work. The primary data of the research is qualitative, as there is only one case study company. We have decided to limit the number of case study companies to one, so that the scope and focus remain steady and can not be distorted. In future research, a quantitative research can validate the method more extensively.

In the case study, influences from outside the closed environment are excluded, as we work in one stable version and deployment of the software product, without following up on updates or changes to the dataset, nor changing the scope of the research subject. This matches the definition of Runeson and Höst (2009), as we investigate the contemporary phenomena in their natural context, without making nor allowing changes to the software product and its environment.

[1] http://www.afas.com

At the case study company, the evolution of the Windows-application to a web-based platform is an actual ongoing process during the execution of the research project. This means that the findings from the cases are also directly applicable to the situation at the case company. A constant process of evolution and validation takes place between the theoretical background and the case study to provide a well formed basis for the envisioned outcome of the research: a generic template method.

2.4 Design science

As this research project aims to compose a generic template method, rather than a situational method, this research is classified as design science (Hevner et al., 2004). The resulting method is designed to be applicable in any case where a software product is to evolve by mapping its functionality between software platforms.

The results of the literature review and the case study cover a variety of subjects that contribute to the composition of the template method. It contains information about how to identify and characterize functionality within a software product, and how to describe the contextual factors that influence whether or not a software product functionality can be mapped on another persona and platform, or not.

2.4.1 Information systems research framework

The image in Figure 2.5 shows the information systems research framework, as proposed by Hevner et al. (2004). It describes the positioning of the research in the environment on the one hand, and the knowledge base on the other hand.

The roles, capabilities and characteristics of the *people* in the environment focus around the field of Software Product Management. The role who instantiates the method is therefore also a software product manager. The capabilities must include making decisions, as this is important in assigning priorities to functionality and mappings. Not having those capabilities would counter the efficiency of the method.

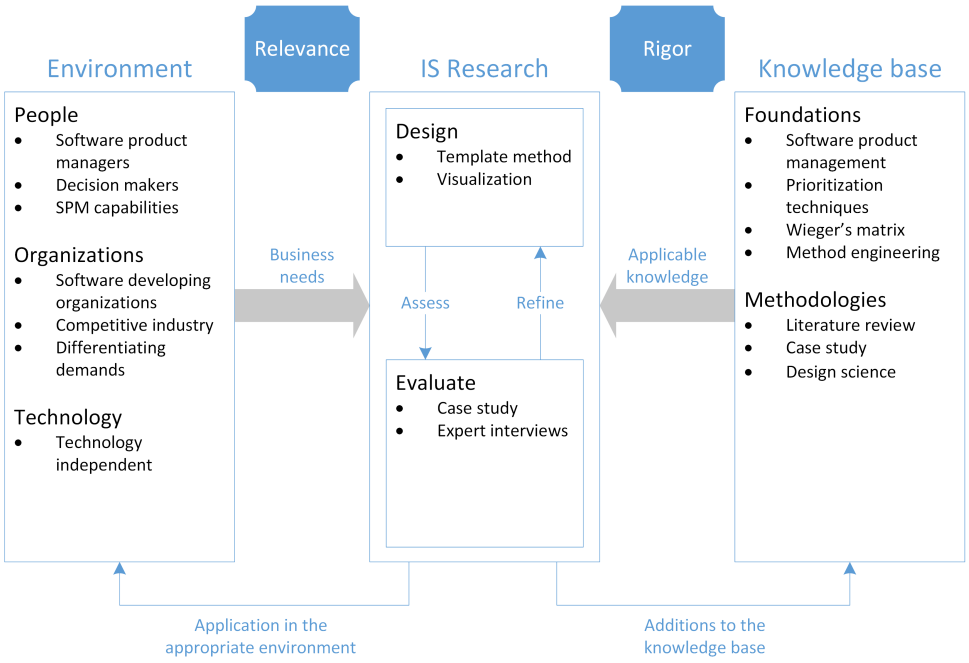


Figure 2.5: Information systems research framework (Hevner et al., 2004)

The *organizations* for which the method is to be designed are software developing organizations. They are active within a competitive industry, which drives them to seek for and act on opportunities, for instance in emerging technologies and next-generation software platforms. Due to the differentiating demands from their market, organizations need a constant and sharp steering of their product roadmap, and thus an efficient way to do so.

The *technology* to which the method applies is technology independent. This is due to the fact that the method focuses especially on the functionality of software, and not the underlying technical or architectural aspects.

The research that is carried out *designs* a template method which serves as a structured approach towards the evolution of a software product. In order to present the outcoming deliverables of the method, a visualization technique has to be designed. However, as depicted in the elaboration of the template method, the correct medium for the reporting of the outcome of an instantiation depends on the selected stakeholders as audience.

The research is *evaluated* through means of multiple case studies and expert interviews. The case studies aim to evaluate the instantiated method by analysis of its performance and to design improvements, which are captured as method increments. The expert interviews apply domain knowledge to the method in order to test it without the direct necessity to go through an entire case study. They also reflect the knowledge of a software product manager on the method, which suits the audience for whom the method is designed.

The knowledge base serves *foundations* of the research through the field of Software Product Management. Here we find methods and techniques, for instance for the identification and prioritization of software functionality, which are included in the research. An example of such a technique is the Wieger's matrix (Wiegers, 1999a). Also, method engineering (Brinkkemper, 1996) serves as a foundation of the research, defining how to design, improve and document a method and rationalize on a meta-level.

Methodologies that are included in the research include that of a literature review, case study and design science. The literature review is inspired by the PRISMA 2009 method of Moher et al. (2009), yet it does not follow its precise method due to unnecessarily produced overhead. The case studies of the research are guided by guidelines of Runeson and Höst (2009) and Yin (2009). Furthermore, the designed method is brought down to a generic level by design science (Hevner et al., 2004), making the method applicable in any situation.

Chapter 3

Requirements Management in Software Product Management

Software Product Management (SPM) concerns “the discipline and role, which governs a product (or solution or service) from its inception to the market/customer delivery in order to generate the biggest possible value to the business” (Ebert, 2007). The SPM Competence Model (Bekkers et al., 2010) in Figure 3.1 gives an overview of the focus areas a software product manager is responsible for, and how these areas are interrelated. The SPM Competence Model includes internal and external stakeholders which (try to) influence the activities. In Van de Weerd et al. (2006), the basis for the SPM Competence Model is laid out, of which we discuss its details in the section below.

On the left side of the model, external stakeholders in the field of Software Product Management (Lehtola et al., 2005) are shown. The *Market* concerns potential customers, competitors and market analysts. *Partners* can be implementation partners, development partners and distribution partners. *Customers* are the current customer base, using the software product and submitting requirements for new releases.

The right side of the model identifies internal stakeholders of Software Product Management (Condon, 2002; Dver, 2003). The operational execution and decision making of these stakeholders is easier to be influenced compared to external stakeholders (Van de Weerd et al., 2006), as they are expected to act in favor of the software product and the vendor organization. The *Company board* defines the corporate strategy, which has to be translated into the strategy, tactics and operations of the product management department. *Research & innovation* explores product innovations and improvements. The internal stakeholders from *Services* implement the software at new or existent customers. New releases are developed and implemented into the system by the *Development* department. Customer support is provided by the *Support* team, which may comprise different layers to separate by level of difficulty and in-depth analysis of the software. *Sales & marketing* is concerned with the first contact with new, potential customers.

The multitude of stakeholders from the roles mentioned before gives a good impression of the complexity of the field of Software Product Management. Each stakeholder role, each organization and each individual has its own agenda, with its own priorities, and all concerning one single software product. In the case of product software, resources are limited, and thus trade-offs need to be made. It is these trade-offs which make requirements management and release planning such complicated competencies.

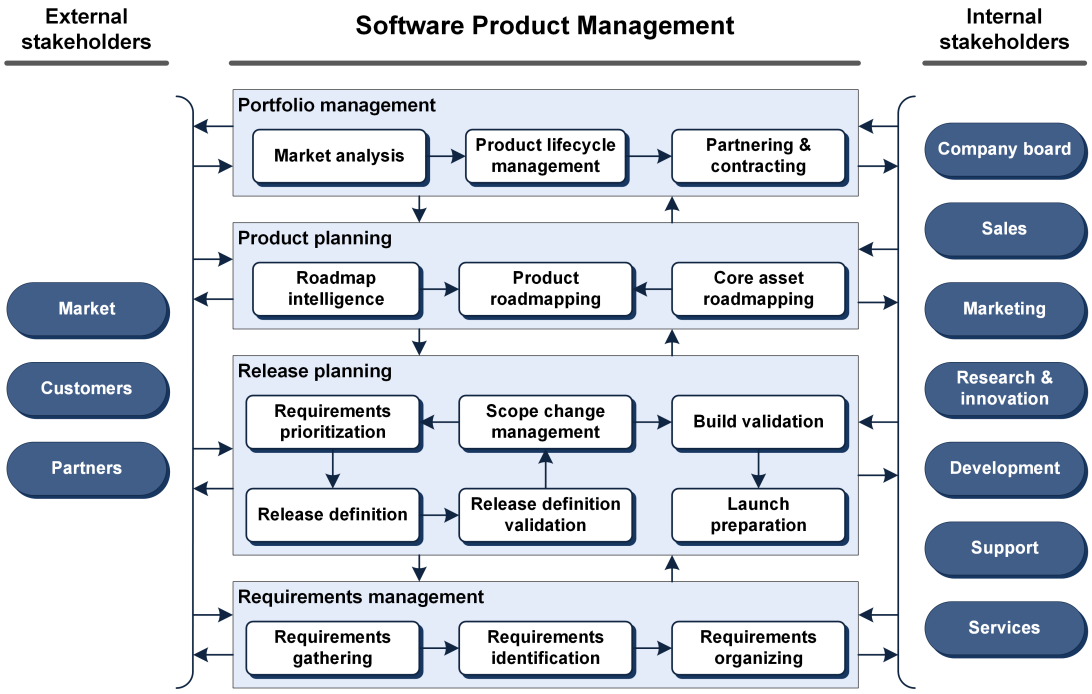


Figure 3.1: Software Product Management Competence Model (Bekkers et al., 2010)

The goal of requirements management is to maintain the requirements continuously throughout the development and system life cycle and thus to ensure that a consistent and up-to-date requirements specification is available at all times (Pohl et al., 2005). For Software Product Management, this means that a software product manager is concerned with requirements management in order to structure them for release planning. This is also visible in the SPM Competence Model in Figure 3.1, where requirements are gathered, identified and organized, prior to being prioritized in the release planning.

As stated in the CHAOS report (The Standish Group, 1995), inadequate requirements management is a major cause of problems in software projects. Given the problem statement in Section 1.1 and the complexity of requirements management in Software Product Management due to a variety of stakeholders interests, we emphasize the importance of efficiency in SPM practices.

Software evolution is an issue of all times, and the rapid pace of technological advancements introduces opportunities which are waiting to be implemented. To support software evolution by the implementation of new software platforms, the Software Functionality Evolution Method introduced in this research attempts to increase the efficiency in the SPM practices. The structured approach allows for comparison and benchmarking of results, which also contributes to synergy in a product management team. To explain how the template method is positioned within the SPM Competence Model, the next section explains the relationships with the various competencies.

3.1 Positioning on the SPM Competence Model

By positioning this research project on the focus areas of the SPM Competence Model (Figure 3.1), we are able to relate the research with the field of Software Product Management, and thus indicate its relevance in the scientific research area of Software Product Management. Figure 3.2 visualizes how the research is related to the Competence Model. In the remainder of this section, we give a brief summary of the focus areas to which this research project relates, and describe how the template method contributes to this aspect of Software Product Management.

Three categories are used for the relationships with focus areas: (1) *triggers* which instantiate the template method, (2) *execution* for the mutual support of the instantiation of activities, and (3) *output* for those focus areas that can use the results of an instantiation for their activities.

***Triggers* focus areas**

Market analysis gathers decision support information about the market needed to make decisions about the product portfolio of an organization (Bekkers et al., 2010). The result of this focus area of Software Product Management reveals new opportunities for software developing organizations, concerning potential software platforms to develop new applications for.

Product lifecycle management concerns the information gathering and key decision making about product life and major product changes across the entire product portfolio (Bekkers et al., 2010). Decisions made in perspective of product lifecycle management serve as input for the template method, and can even cause an instantiation of the template method.

***Execution* focus areas**

Requirements gathering concerns the acquisition of requirements from both internal and external stakeholders (Bekkers et al., 2010). The template method which is designed in this research project incorporates the identification of functionality in an existent software product. This functionality serves as input for requirements of the application on another software platform.

Requirements organizing structures the requirements throughout their entire lifecycle based on shared aspects, and describes the dependencies between product requirements (Bekkers et al., 2010). Since functionality is grouped by the entity it acts upon, clusters of functionality can be formed, which indicates a certain degree of dependency between sets of functionality.

Requirements prioritization prioritizes the identified and organized requirements (Bekkers et al., 2010). Different prioritization techniques are described as part of the template method, of which the choice depends on the complexity of the project. Mapping occurs between a scenario of a persona on a software platform and a set of functionality. To this "requirement", a priority is assigned.

***Output* focus areas**

Release definition selects the requirements that will be implemented in the next release, based on the prioritization they received in the preceding process. It also creates a release definition based on the selection (Bekkers et al., 2010). The results of an instantiation of the template method can be used to create a release definition. The prioritized list of mappings and their interdependencies assist in the selection of requirements for a future release.

Roadmap intelligence gathers decision supporting information needed in the creation of the product roadmap (Bekkers et al., 2010). The output of an instantiation of the template method, which is a list of ordered requirements for a new software application on another software platform, inspires themes and central subjects for upcoming roadmaps.

Product roadmapping deals with the actual creation of the product roadmap itself (Bekkers et al., 2010). The prioritized list of requirements, produced by an instantiation of the template method, can be used as input for the requirements and themes to put on a roadmap. Obviously, those requirements with the highest priority are candidates for an implementation on the short term, while those with a lower priority can be postponed to later releases.

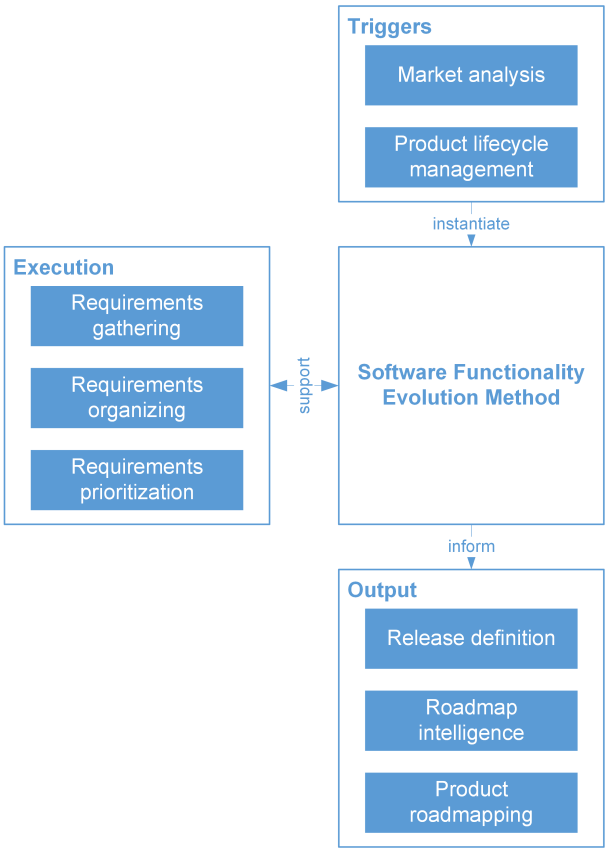


Figure 3.2: Positioning of the research on the SPM Competence Model

Chapter 4

The Software Functionality Evolution Method

This chapter describes the final version of the template method that was designed during the course of this research project. The process of the development of the template method and its evolution through analysis of its performance in multiple case studies is described in Chapter 5. This chapter starts with a description of method engineering, and its application in this research. The Process-Deliverable Diagram of the template method is presented in Section 4.2. The diagram's corresponding activity and concept tables are described in respectively Appendix B and Appendix C. The remainder of this chapter elaborates on the five main phases of the method, and the theoretical foundations that explain the phases.

4.1 Method engineering

Method engineering is "the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems" (Brinkkemper, 1996). In Van der Schuur (2011), the term "template method" is introduced. A template method differentiates from situational method engineering by the fact that it prescribes *what*, rather than *how*, activities and concepts are to be instantiated in case of a template method instantiation (Van der Schuur, 2011). By applying this view on method engineering, this research project introduces a template method (rather than a situational method) which can be instantiated in different cases with different project requirements. This varies from the situational method engineering approach, where the method is designed to be tuned for one situational case.

The template method designed in this research project is instantiated in multiple case studies. The analysis of the performance of the template method instantiation enables the improvement of the template method, by designing method increments. In Van de Weerd et al. (2007), the authors mention that a definition of a method increment seems not to be available, and therefore a definition is proposed: a method increment is "a method adaption, in order to improve the overall performance of a method" (Van de Weerd et al., 2007). Based on the analysis of earlier case studies and the structure of the meta-meta-model of a Process-Deliverable Diagram, Van de Weerd et al. distinguish that 18 elementary types of method increments can be distinguished:

- *insertion* of a concept, property, relationship, activity node, transition, role
- *modification* of a concept, property, relationship, activity node, transition, role
- *deletion* of a concept, property, relationship, activity node, transition, role

In this research project, we are inspired by the work of Van der Schuur (2011) to apply constructivist hermeneutics in information systems research in order to reflect on the process of the research. To reflect on the process of method improvement through method increments, we propose another categorization of method increments, other than that of Van de Weerd et al. (2007), which is further elaborated in Chapter 5. The correct application of the categorization allows for more transparent reflection on incremental method engineering.

4.2 Process-Deliverable Diagram

In Figure 4.1, the Process-Deliverable Diagram (PDD) of the Software Functionality Evolution Method (SFEM) is shown. A PDD is a meta-modeling technique used for modeling activities and artifacts of a certain process (Van de Weerd and Brinkkemper, 2008). In the PDD, *activities* resemble a method's process, of which the deliverables are represented as *concepts*. Both activities as concepts know a simple and complex form, of which the complex form consists of a collection of sub-elements. A complex activity or concept can be open or closed, depending whether or not the sub-elements are known and relevant in the context of the PDD.

On the left side of the diagram are the *activities* of the method's process, of which the notation is based on the UML activity diagram (Object Management Group, 2004). These activities are to be implemented in the practices of the software developing organization, in order to be able to increase efficiency in the Software Product Management practices concerning software evolution. As in method engineering, we make a distinction between activities and main activities. Different relationships between activities exist, which are sequential, unordered, concurrent and conditional. In the SFEM, main activities resemble phases of the template method. The method's five phases cluster the variety of activities around a certain subject. As from Section 4.3 onward, the main activities of the method are further explained. A complete overview of the SFEM's activities is given in Appendix B.

On the right side of the diagram, deliverables are visualized as *concepts* to indicate what artifacts are produced by a template method instantiation, of which the notation is based on the UML class diagram (Object Management Group, 2004). Different relationships between concepts exist, which are generalization, association, multiplicity and aggregation. In the PDD, we have visualized aggregations between concepts with a blue line instead of the common black color. We have decided to do so in order to improve the comprehensibility of the diagram and method, as the diagram became obscured by the many lines between concepts. However, this is not according to the official technique, as defined by Van de Weerd and Brinkkemper (2008). A complete overview of the SFEM's concepts is given in Appendix C.

The Software Functionality Evolution Method consists of five main phases, which are depicted as main activities:

- *Project definition*, exploring the basis of the project (Section 4.3)
- *Functionality identification*, extracting entities and functionality from the software (Section 4.4)
- *Scenario creation*, identifying scenarios of personas on software platforms (Section 4.5)
- *Functionality mapping*, assigning a priority to the mapping of functionality on scenarios (Section 4.6)
- *Results reporting*, reporting the results of an instantiation to the project stakeholders (Section 4.7)

The remainder of this chapter will elaborate on the activities and concepts within the main activities of the template method.

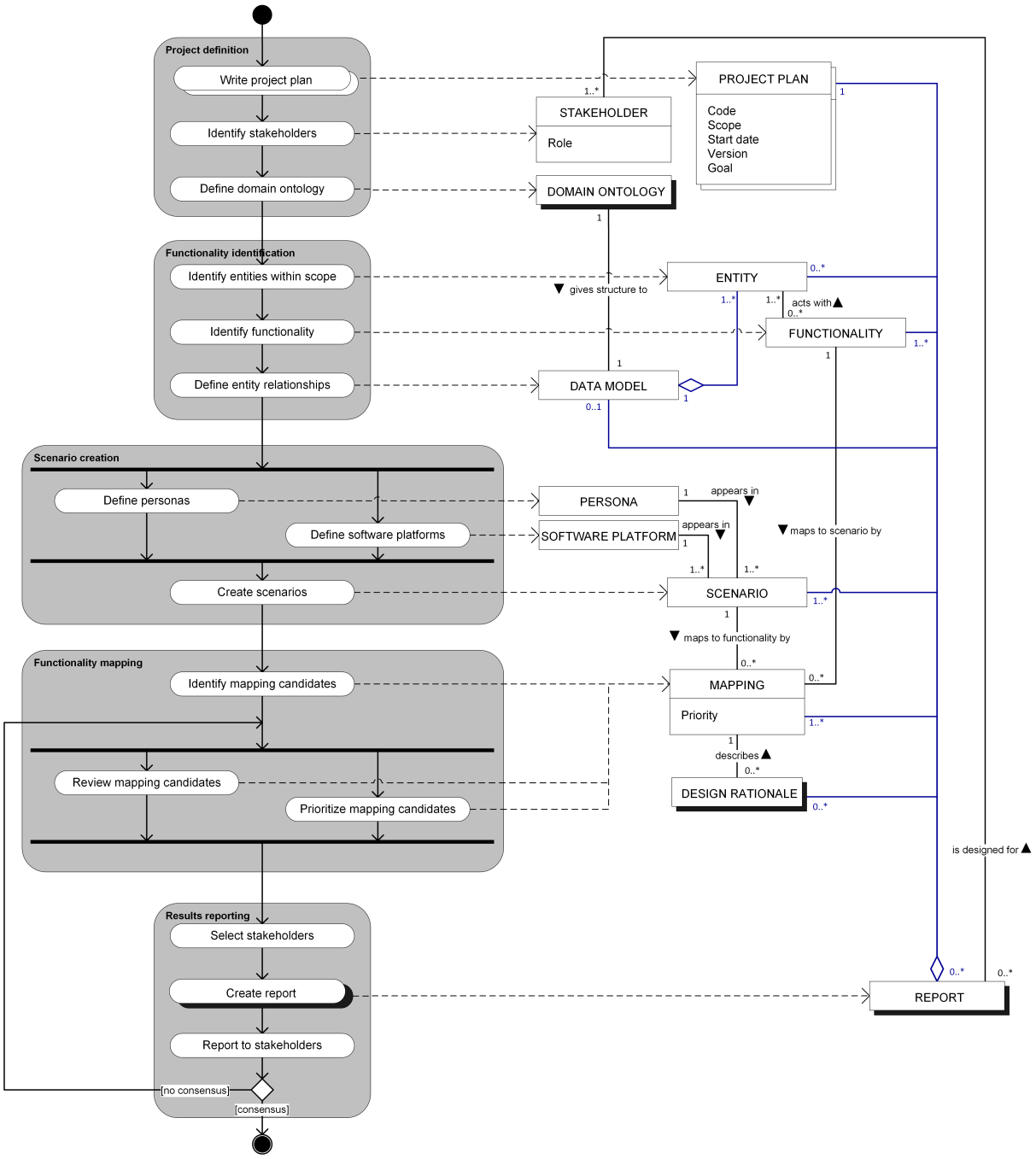


Figure 4.1: Process-Deliverable Diagram of the SFEM

4.3 Project definition

To start an instantiation of the template method, the project has to be defined and prepared. The basis for the rest of the template method and its instantiations is created by instantiating the activities and concepts of the main activity *Project definition*.

At first, an instantiation of the concept PROJECT PLAN has to be created. The concept is a formal definition of the metadata of the project, created by the activity *Write project plan*. The activity is a complex open activity, because it's known that the sub activities are centered around the properties of the concept PROJECT PLAN.

Second, the relevant stakeholders of a project are identified by instantiating the activity *Identify stakeholders* and by instantiating the concept STAKEHOLDER at least once. Each stakeholder has a role, which can be identified by the Method Stakeholder Classification Matrix (Section 4.3.1). An instantiation of the concept REPORT is designed specifically for one or more stakeholders, and stakeholders can be included in the group session during activities of the *Functionality mapping* main activity.

Last, the ontology of the domain of discourse is captured in an instantiation of the concept DOMAIN ONTOLOGY. This concept plays a role in the next main activity, centered around the extraction of entities and functionality from the system.

4.3.1 Method Stakeholder Classification Matrix

In method engineering, it is possible that a process is explicitly carried out by a specific individual or organizational role. In that case, the role is indicated in the activity depicted in the method (Van de Weerd and Brinkkemper, 2008). The usage of roles is not required in method engineering, and a role can be applicable to both activities and sub-activities.

Roles depict the actual individual or organizational role who is responsible for carrying out a specific activity. On the other hand, stakeholders are involved in the method instantiation to provide the necessary input. There are also stakeholders involved who share an interest in the output of the method, the instantiated concepts. To help identify these stakeholders, and apply a classification to their role in the method's instantiation, we introduce the Method Stakeholder Classification Matrix (MSCM), visualized in Figure 4.2.

Degree of participation The *degree of participation* defines how active a stakeholder is within the instantiation of the method. Active stakeholders play a role during the length of the method instantiation, while passive stakeholders are only interested in a part, or only just its deliverables.

Degree of interaction The *degree of interaction* defines how direct a stakeholder acts with the deliverables of the method. A direct relationship implies that the stakeholder gets the results of the method directly through its deliverables, or even contributes to the creation of the deliverables. An indirect relationship implies that the stakeholder only sees results of the method in an edited, more focused way, without being aware of the decisions or precise details.

		Degree of participation	
		Active	Passive
Degree of interaction	Direct	**Participant** - Project manager - Product manager - Domain expert	**Observer** - Project manager - Product manager - Board member
	Indirect	**Informer** - Domain expert - Customer - Partner	**Outsider** - Developer - Designer

Figure 4.2: Method Stakeholder Classification Matrix

Participant The stakeholder role *Participant* is an active stakeholder who directly interacts with the deliverables of the method. The stakeholder can be involved as the individual who instantiates the method, or someone who delivers direct input through means of expert validations or interactions based on questions of other method stakeholders.

Observer The stakeholder role *Observer* is a passive stakeholder who directly interacts with the deliverables of the method. This could be someone who needs to approve or understand the output of the method, without it being edited by stakeholders other than those with the role Participant.

Informer The stakeholder role *Informer* is an active stakeholer who indirectly interacts with the deliverables of the method. This means that in order for the stakeholder to interact with the deliverables, a passthrough from a direct stakeholder is necessary. The Informer often acts as a provider of knowledge for the Participant stakeholder role.

Outsider The stakeholder role *Outsider* is a passive stakeholder who indirectly interacts with the deliverables of the method. The stakeholder plays no role in the method instantiation process and does not get to see the whole picture of the method's deliverables, but is rather provided with a modified view with an as-is status.

4.3.2 Domain ontology

An ontology describes the entities within the domain in discourse, and how these entities are interrelated (Gruber, 1993). The domain of discourse is the environment in which the software product in scope is designed to operate. The template method is designed for the evolution of a software product which is currently operational, which means that a domain ontology is scoped at the operational environment.

The identification of different abstraction levels of entities and groupings in the software product assists in the scoping of the project. This enables an analyst to create an overview of the software architecture, and discuss at different levels of detail with stakeholders. A domain ontology lays the basis for the further identification of entities and their functionality within the domain of discourse.

4.3.3 Activity and concept table

Table 4.1 gives a summary of the activities within the main activity *Project definition* of the Software Functionality Evolution Method.

Main activity	**Sub activity**	**Description**
Project definition	Write project plan	The project's outline is defined, so that a common understanding of the project's goal and properties is set. This concept is also included in the REPORT. The property *Scope* of the concept limits the exploration of ENTITIES and FUNCTIONALITY in the project instantiation.
	Identify stakeholders	Each STAKEHOLDER of the project is identified and his/her *Role* in the project is noted. The person who instantiates the template method is also a STAKEHOLDER, as are STAKEHOLDERS who are only interested in the final REPORT. In Section 4.3.1, a Method Stakeholder Classification Matrix is proposed, which can help in the identification and labeling of the *Role* of the STAKEHOLDER.
	Define domain ontology	By analyzing the ENTITIES within the domain of discourse and how those ENTITIES are related, a DOMAIN ONTOLOGY can be defined (Gruber, 1993). The underlying descriptive models of the software product can assist in retrieving an accurate description of the DOMAIN ONTOLOGY. The DOMAIN ONTOLOGY lays a basis for the DATA MODEL.

Table 4.1: Activity table of the SFEM,
main activity *Project definition*

Table 4.2 gives a summary of the concepts that are instantiated by the activities within the main activity *Project definition* of the Software Functionality Evolution Method.

Concept	Description
PROJECT PLAN	The PROJECT PLAN is a document that describes the technical and management approach to be followed for a project. The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way that the project will be organized (IEEE Std. 610.12-1990). The PROJECT PLAN is included in the REPORT of the project's results, and in any other documentation that acts as a deliverable of the project. The concept's properties *Scope* and *Goal* play an important role in the further instantiation of the template method.
STAKEHOLDER	A STAKEHOLDER is an individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations (IEEE Std. 12207-2008). The STAKEHOLDER has a predefined *Role* in the project. At least one STAKEHOLDER is the project manager, who is the person with overall responsibility for the management and running of a project (ISO/IEC/IEEE 26512). A STAKEHOLDER's role can be organized as described in Section 4.3.1.
DOMAIN ONTOLOGY	An ontology describes the ENTITIES within the domain in discourse, and how these ENTITIES are interrelated (Gruber, 1993). The DOMAIN ONTOLOGY represents the domain in which the software product is designed to operate, the domain of discourse. It is composed of higher-level ENTITIES which are identified in the product's functional architecture. The DOMAIN ONTOLOGY lays the basis for the DATA MODEL.

Table 4.2: Concept table of the SFEM,
main activity *Project definition*

4.4 Functionality identification

By extracting entities and functionality from the software product, an overview of the current capabilities of the system is created. Because a set of functionality can influence one or more entities, the relationships between entities become evident, which establishes an understanding of the system's data model.

The main activity starts with identifying all entities within the scope of the project. This creates instantiations of the concept ENTITY. These entities are included in the instantiation of the concept DATA MODEL, in which functionality establishes relationships between entities. To identify instantiations of the concepts ENTITY and FUNCTIONALITY, the remainder of this section introduces techniques to assist in the identification of the concepts, and improve efficiency.

The activity *Define entity relationships* creates an instantiation of the concept DATA MODEL, which may play a role in the concept REPORT, and explicates the way entities are linked together by functionality in the software product. A data model can be represented by a UML class diagram (Booch et al., 1999), but a simple tree diagram can also suffice, depending on the complexity of the software product at hand.

It is possible that new relevant entities and functionality are discovered during the further instantiation of the template method, making the project's progress return to the main activity *Functionality identification*. This is not wrong, as it contributes to a more complete view of the software product. However, we have decided not to explicitly design this transition, because returning to the main activity *Functionality identification* may occur from any activity in the template method.

4.4.1 Software functionality identification

In order to compose the template method which is to be designed in this research project, we have performed research to find a structured approach to identifying and characterizing functionality in a software product. However, since the method which is to be designed in this software product is designed for software product managers, an in-depth analysis of source code and data representations in databases does not suit the projected audience of this research deliverable. Therefore, the described techniques focus on a functional, higher level view of software functionality, instead of a technical or architectural level of detail.

User manual The functionality in a software product can be identified by many non-technical approaches. Well-documented software with an extensive user manual serves as an excellent source for analysis of its architecture and features. Documentation is often arranged in processes instead of single features, because it focuses on the goals of the software's user. This helps an analyst by thinking in processes, which is exactly how a user would behave as well.

A user manual's items often have hyperlinks which link to other relevant content, which creates a navigation structure of related items. By following hyperlinks, items related to the project's scope can be identified, followed by further analysis if a feature is indeed relevant. An advantage of this approach is that software functionality can be identified, without becoming too distracted by the interaction with the user interface of the software product. The analysis of a combination of textual and visual descriptions in a user manual gives a good overview of the required input and the goal that is achieved by the functionality.

A possible downside to this approach is that user manuals can be outdated, since documentation often follows up on updates of software. Secondly, it is uncertain if all relevant functionality of the software product is described in its documentation.

In Royce (1970), the importance of documentation throughout the entire process of software development is stressed, from design to development, testing and being operational. It enables communication between software stakeholders, such as designers, developers, management and customers, without the need of direct interaction with one another.

Graphical user interface Another functional approach to software product functionality identification is the analysis of interaction with the software product's graphical user interface (GUI). If a user interface is present, it allows for manual and interpretive analysis of the software's functionality.

An advantage of this approach is that the user interface is supposed to display the available functionality, as it would not be functional if it could not be triggered by a user. By interacting with the user interface, processes can be taken as starting point of the analysis. These processes resemble the daily activities of potential users of the product. For instance, a single process could require multiple input actions of a user, meaning that a single process could span multiple sets of functionality.

A downside to this approach is that the user must have an environment of the software where all functionality is available, thus not restricted by authorization or disabled views or interfaces. Furthermore, the approach is more error-prone because functionality may not be revealed when certain scenarios are omitted, for instance when it is dependent on the data inserted by the user.

Formal design artifacts If formal design artifacts are available in the software developing organization, such documents can also assist in the identification of software product functionality. These design descriptions can be both at a technical or functional level of detail.

As described in the UML specification (Booch et al., 1999), structure diagrams are more focused on technical aspects of modeling, such as data representations and architectural descriptions, while behavior diagrams concern the functionality that a system is designed to perform. While this means that behavior diagrams are most suitable for the identification of processes, workflows and functionality in software products, structure diagrams can also assist the identification of entities, which can then be used for analysis of functionality.

Since design artifacts can reveal the relationships between entities, it enables an analyst to identify a large portion of the possible functionality, since those relationships also need to be maintained by functionality.

Architecture reconstruction Architecture reconstruction is defined by Kazman et al. (2003) as "the process where the 'as-built' architecture of an implemented system is obtained from an existing legacy system". Tools can assist in the extraction of information from an existent system, and generating visualizations at different levels of abstraction. The authors introduce ARMIN, the Architecture Reconstruction and Mining tool, which is a follow-up of the Dali Architecture Reconstruction workbench (Kazman and Carrière, 1999). Kazman et al. identify a software architecture reconstruction process which comprises five phases:

1. Information extraction
2. Database construction
3. View fusion
4. Architectural view composition
5. Architecture analysis

The report of Kazman et al. (2003) mentions different types of tools to be used in the extraction of architectural information, each being applied to the raw source code of a software product. Considering the goal of this research project, the abstract syntax tree (AST) analysis is a relevant approach, as it builds an explicit tree of the information about the architecture of the software product. Also, lexical analyzers can help to identify the relationships between entities in a software product, and thus reconstruct its architecture.

The architectural view composition phase is of particular interest for this research project, as it helps in creating a visualization of the software architecture, and reveal the relationships between entities. This allows for the final phase, the architecture analysis phase, to help an analyst in retrieving information about the software product and the entities and functionality incorporated.

In O'Brien et al. (2002), a categorization of approaches and tools for architecture reconstruction is given, including manual architecture reconstruction, manual reconstruction with tool support, query languages for writing patterns to build aggregations automatically and other techniques, such as clustering, data mining and architecture description languages (O'Brien et al., 2002).

A popular example of a manual reconstruction method with tool support is Rigi (Müller and Klashinsky, 1988). The Rigi Standard Format (RSF) is the file format which represents the information that was extracted by the parser. By manual processing of the information retrieved from the software product, groupings, collapsing and filtering creates a visualization suitable for the project needs.

An example of a query language is the Dali workbench (Kazman and Carrière, 1999), which is also mentioned in the reports by Kazman et al. (2003) and O'Brien et al. (2002). The workbench includes the PostgreSQL database system and is an extension to the Rigi tool. The extension generates different architectural views of the system, based on the information extracted from the system by the Rigi tool. Queries sent to the PostgreSQL database in the form of Structured Query Language (SQL) enable an analyst to generate the architectural views.

Natural Language Processing Natural Language Processing (NLP) is "an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things" (Chowdhury, 2003). Through means of techniques such as lexical analysis and text segmentation, large sets of text can be analyzed in an automated manner, in order to make conclusions about the actual contents of the text. Tokenization implies creating "tokens", sets of characters, words, phrases or other elements, based on elements retrieved from a text.

In the analysis of a software product's architecture, in order to identify entities and their functionality, NLP and related techniques can help an analyst to analyze nearly any textual description of the software's architecture. This includes user manuals, formal design artifacts, translation files, and changelogs.

The textual analysis can result in different forms. For instance, tag clouds (or word clouds) can help to visualize the relative occurrence of a token by differentiating in size, dependent on the occurrence compared to other token. Furthermore, analysis of the text can result in linking of tokens because of their similar occurrence.

Based on the multitude of possibilities by application of Natural Language Processing techniques, not only does it help in the identification of functionality within a software product, but also does it help in the identification of entities within a software product, and their interrelationships.

4.4.2 Software functionality classification

The classification of software functionality helps to make a distinction between different sets of functionality. In the case of a software system with a large set of functionality, a classification gives an overview of the larger whole. The classification does not only help to better understand a set of functionality compared to the larger whole, but it also helps to go deeper into a set of functionality and understand what it concerns.

Applying a classification to functionality also helps in the further identification of functionality. For instance, if one is able to add and view a certain entity, such as a customer, one should also be able to modify and remove the customer entity. Although this reveals a relatively basic set of functionality, it does speed up the identification process.

The role of a classification proves to be relatively useful, as is exemplified in the case studies of this research project. By means of a scenario of a persona on a software platform, one classification may prove to be more important than the other. For instance, a scenario could reveal that creating an entity is more important than deleting one, and viewing an entity is more important than creating one. A classification helps to be more efficient in the mapping of functionality, based on their classification, as all deletions may receive a lower priority by default.

CRUD One of the most common and basic classifiers of functionality is the CRUD classification (Martin, 1983). CRUD stands for *Create*, *Read*, *Update* and *Delete*. The classification depicts the type of action that is performed on a certain entity. When seen from the entity-perspective, any action to be performed on the entity can be classified with the CRUD classification. This classification can be compared to the *Insert*, *Select*, *Update* and *Delete* operators which are present in SQL.

sCRUD and BREAD In the research by Cooper et al. (2010) it has been proposed to extend the CRUD classification with an extra *Scan* operator, which helps to make a distinction between reading a single record or scanning multiple records in a specific order. If applicable, this extension can be included to make a better distinction between reading a single or multiple records at once, and thus seeing only a summary of a record (*scan*), or seeing all the details of a record (*read*). Stolze et al. (2007) explicitly specify the BREAD classification as part of their workflow task model. BREAD is an abbreviation of *Browse*, *Read*, *Edit*, *Add* and *Delete*. As in sCRUD, the *Browse* operation is an extra operator which can be used to read more than one entity.

Read-only / **Maintain** Derived from authorization levels in the software product of the case studies, functionality can also be classified according to two operators: *Read-only* and *Maintain*. This is a noteworthy classification, because it is common that when you would authorize the functionality to create an object, one would also need to be able to update and delete the same object. Therefore, merging the *Create*, *Update* and *Delete* operators under one option reduces the amount of functionality to analyze and thus increases efficiency.

One should proceed with caution before deciding on merging these operators, and perform an extensive check first to make sure that there is no harm done by merging these operators. For instance, if the project circumstances depict that creating and updating an object in a scenario is preferable, yet deleting the object is a back-office task which is not to be present in the current scenario, using the *Read-only* / *Maintain* operators would not be suitable.

4.4.3 Activity and concept table

Table 4.3 gives a summary of the activities within the main activity *Functionality identification* of the Software Functionality Evolution Method.

Main activity	Sub activity	Description
Functionality identification	Identify entities within scope	Limited by the *Scope* of the PROJECT DEFINITION, the ENTITIES within the project instantiation's *Scope* are identified. The underlying descriptive models of the software product can help in the identification of ENTITIES.
	Identify functionality	Based on identified the ENTITIES and an analysis of the software product in scope, FUNCTIONALITY is identified and linked to the ENTITIES it corresponds with. Section 4.4.1 describes different methods on how to identify FUNCTIONALITY from a functional perspective. FUNCTIONALITY can be organized by a functionality classifier, as described in Section 4.4.2, which makes identification of all FUNCTIONALITY within the project scope easier and more efficient.
	Define entity relationships	By defining the relationships amongst ENTITIES, the concept DATA MODEL is instantiated. The relationships can be extracted from underlying descriptive models, and from descriptions of FUNCTIONALITY, by interpreting on which ENTITIES the FUNCTIONALITY acts.

Table 4.3: Activity table of the SFEM, main activity *Functionality identification*

Table 4.4 gives a summary of the concepts that are instantiated by the activities within the main activity *Functionality identification* of the Software Functionality Evolution Method.

Concept	Description
ENTITY	In computer programming, an ENTITY is any item that can be named or denoted in a program. For example, a data item, program statement, or subprogram. (IEEE Std. 610.12-1990). The concept FUNCTIONALITY is applicable to ENTITIES, and the relationships between ENTITIES are represented in the DATA MODEL.
FUNCTIONALITY	FUNCTIONALITY concerns the capabilities of the various computational, user interface, input, output, data management, and other features provided by a product (IEEE Std. 1362-1998). Techniques to identify software FUNCTIONALITY have been described in Section 4.4.1. To organize the instances of FUNCTIONALITY, a classification can be applied, as is described in Section 4.4.2.
DATA MODEL	A DATA MODEL identifies the entities, domains (attributes), and relationships (associations) with other data and provides the conceptual view of the data and the relationships among data (IEEE Std. 1320.2-1998). The DATA MODEL serves as a basis for the remainder of a template method instantiation and can be included in a REPORT. A DATA MODEL can be represented by a UML class diagram (Booch et al., 1999), but a simple tree diagram can also suffice, depending on the complexity of the software product at hand.

Table 4.4: Concept table of the SFEM,
main activity *Functionality identification*

4.5 Scenario creation

When the characteristics of the functionality of the software product in scope have been defined, the project may continue to explore other factors which may influence the mapping of functionality between software platforms. We limit the influential factors to constraints raised by the users of the system, and the software platform we are mapping functionality on. We define the possible appearance of a user persona on a given software platform as a *scenario*.

In the SFEM, a scenario represents the possible occurrence of a persona on a given software platform. It is not to be confused with the common definition of a scenario in the field of software engineering, which is “a description of a series of events that may occur concurrently or sequentially” (IEEE Std. 829-2008), as the scenarios in the template method do not represent a specific sequence of events.

In order to create instantiations of the concept SCENARIO, the template method explores relevant personas and software platforms that (may) play a role in the software product’s evolution. The emphasis here lays on the word *relevant*, as many personas and software platforms may be identified for even a single software product. The same goes for the instantiations of the concept SCENARIO, where we only instantiate the concept if it would be relevant and natural that the given persona uses the given software platform.

The activities *Define personas* and *Define software platforms* are parallel activities, since the order of the activities is not strict, and instantiations of one concept may result in instantiations of the other. Both an instantiation of PERSONA as SOFTWARE PLATFORM appear in at least one instantiation of SCENARIO, otherwise it is not relevant to the project, and thus the instantiated concept would need to be excluded.

4.5.1 Persona

A persona is defined as a social role of a person in a specific context (Jung and Storr, 1983). However, as suggested in Aoyama (2005), there is no strict guideline to identifying personas, which is why Aoyama uses a rational approach to identify personas.

Given the context of the template method, a persona thus represents a (potential) user of the system. It is the definition of the persona which represent its ability or inability to execute certain functionality, and the goals the persona aims to achieve by using the software product. For the mapping of functionality, this depicts the relevance of the functionality in a given scenario.

Cooper (1999) states that an actual user of a system is a valuable resource, yet it is not desirable to let an actual user directly influence the designing process. Therefore, the use of pretend users in terms of personas is a good way to represent the actual users of a system in the designing process. Cooper (1999) therefore defines personas as hypothetical archetypes of actual users, which are defined by their goals.

The research of Junior and Filgueiras (2005) gives a good comparison of user modeling techniques, including user roles, user profiles, user segments, extreme characters and personas. Junior and Filgueiras emphasize the importance of personas, as it is a very concrete representation of an expected user, which leaves little room for interpretation.

Cooper (1999) also suggests defining a persona as specific as possible, which implies giving the persona a real name. This makes the persona become real in the minds of the designers, which makes storytelling more efficient. The feeling of defining a living creature can be amplified by also assigning a photo to the persona, so that it also gets a face and really starts “living”. This personification of the persona is also described by Junior and Filgueiras (2005), who give examples of information concerning personal, technical, relational and opinion aspects.

When designing an interactive system, a product requires a small set of personas rather than just one (Rogers et al., 2011). Though it is hard to recommend how many personas are best, it is recommended to identify a primary persona, which represents a large section of the intended user group.

Based on the literature by Jung and Storr, Aoyama, Cooper and Junior and Filgueiras, the following subsections proposes focus areas which help to define personas. It is not strict that each of the areas should be defined for each persona in every project, as the focus areas are merely guidelines for addressing the definition of a persona.

Characteristics By addressing a persona with a name and giving him/her a face by means of a photo, the persona starts to live in the minds of designers. The characteristics can be extended by demographic data about the user, which add as much facts and details as possible. The more detailed a persona is, the less room is left for interpretation. A too shallow description of the persona creates a so-called *elastic user* (Cooper, 1999), which is a too vague representation of the actual persona. Examples of demographic data that can be included, as well, are gender, age, ethnicity, marital status, career path, spoken languages, and location.

We believe the description of attitudes is also part of the characteristics of the persona. An example is a negative and cautious attitude towards storing data in the cloud, due to privacy concerns. Such attitudes also tell us more about the characteristics of the persona, and the way the persona thinks about certain (relevant) subjects. In Pruitt and Grudin (2003), technology attributes are included in a foundation document to explicate a person’s perspective on technology, past and future. Also, in Sommerville (2007), an attitude defines what is and is not acceptable for users, based on their attitude towards a certain topic.

Needs and goals The defining of goals for personas is fundamental in the designing of interactive systems. There is no use to defining personas if we have no clear image of what they are actually trying to achieve. It is not their actions (or interactions) that are of direct interest, it is what they wish to accomplish which drives the design of the system. Their actions are merely the way they try to achieve their goals, which is something which can be changed and improved by design. As actions are driven by goals, goals can in turn be driven by needs. If a user has a certain need, his/her goal is to satisfy that need. Examples of such needs are the need for information about the age of a co-worker, or the need to create a new invoice for a customer, based on a completed transaction.

Cooper et al. (2012) describes an interesting view on the classification of persona goals, which are *life goals*, *experience goals* and *purpose goals*. Life goals represent personal aspirations, yet typically go beyond the context of the product being designed. Experience goals concern the feeling that arises when using a product. Last, purpose goals describe "what" the user would like to use in a well designed product or service (Junior and Filgueiras, 2005).

Skills and competencies The competencies of a persona give the designers a good interpretation of what a pretend user is capable of doing, or not. Competencies are a great guideline for designers to design functionality, as it defines the fundamentals that enable or disable a user to perform certain actions.

The difference between competencies and skills is that a skill is something learned in order to be able to carry out certain actions, while a competency can incorporate a skill, yet also concern abilities, behaviors and knowledge that are fundamental in order to be able to perform a certain skill[1].

The example we wish to sketch is the ability to use the Command Line Interface (CLI) of a software product. The actual fact to be able to use this CLI or not, is a skill. However, in order to be able to use the CLI, one has to understand the necessity of correct syntax, and thus be precise and flawless in the composing of commands to enter into the CLI, which is a competency.

The driving forces behind skills and competencies are experience and knowledge. Even though skills, competencies, experience and knowledge are driving forces behind one another, the identification of experience and knowledge for a persona also helps to understand what the persona would be able to do.

Constraints Last, but certainly not least, we emphasize the importance of identifying constraints of a persona, and especially the importance of (re-)defining them separately. Constraints may result from personal characteristics, needs and goals, or skills and competencies, yet the explication of persona constraints is of high importance for the mapping of software functionality, and thus requires extra emphasis. It is not only a persona's ability, but also the inability, which helps in deciding whether or not to map software functionality to another software platform.

4.5.2 Software platform

In a publication by Van Angeren (2013), definitions for the term *platform* by Robertson and Ulrich (1998), Bresnahan and Greenstein (1999) and Baldwin and Woodard (2009) are compared. The author identifies common elements from the definitions, which are (1) a certain core of stable components and (2) reuse or extension by other components or parties. The definition of a platform by Gawer and Cusumano (2003), covers those two elements best: "[a platform is] a foundation technology or set of components used beyond a single firm that brings together multiple parties for a common purpose or recurring problem".

[1] Source: http://www.talentalign.com/skills-vs-competencies-whats-the-difference/

Platform type The research by Bosch (2009) gives a two-dimensional typology of software ecosystem taxonomies, where the platform is classified as either *desktop*, *web* or *mobile*. This classification of platforms helps to categorize software platforms on a high-level, without going too much into details about the specifics of the platform. With a view on actual next-generation software platforms and emerging technologies, we extend this classification with a *wearable* platform, which aims to position or contextualize the computer in such a way that the human and computer are inextricably intertwined, so as to achieve Humanistic Intelligence (Mann, 1997). Based on Bosch (2009) and Mann (1997), we propose the following classification of software platforms:

- **Desktop** – A platform which facilitates software on a fixed location, as it requires the installation of and interaction with other components and peripherals.
- **Web** – A platform which does not require components that are not incorporated by default in modern web browsers, and can thus run its software anywhere, at any time.
- **Mobile** – A platform which is suitable for smaller devices that are as mobile as its user.
- **Wearable** – A platform which is attached to the user's body through means of accessories or clothing, and can therefore interact with the user, without the necessity of user input.

Another dimension we add to the classification of a platform type, is the typology of platforms by Gawer (2009), where the author distinguishes four types:

- **Internal platform** – A set of subsystems and interfaces to form a common structure from which a stream derivative products can be efficiently developed and produced. (Muffatto and Roveda, 2002)
- **Supply chain platform** – A set of subsystems and interfaces that forms a common structure from which a stream of derivative products can be efficiently developed and produced by partners along a supply chain. (Gawer, 2009)
- **Industry platform** – A product, service or technology, that is developed by one or several firms, that serves as a foundation upon which other firms can build complementary products, services or technologies. (Gawer, 2009)
- **Two-sided market** – A market in which a platform facilitates transactions between at least two distinct groups of actors. (Rochet and Tirole, 2003; Rysman, 2009)

SWOT analysis One way to analyze the functional characteristics of a software platform is to perform a SWOT analysis. SWOT stands for *Strengths*, *Weaknesses*, *Opportunities* and *Threats*. Strengths and weaknesses concern factors with an internal origin, so from within the software platform. Opportunities and threats concern factors with an external origin, so from the software platform's environment. Although the analysis was originally designed to analyze an organization and its environment, it can just as well be applied to any other situation that can generate a competitive advantage for an organization. Based on the results of a SWOT analysis, the strategy is to build on your strengths, minimize your weaknesses, seize opportunities and counteract threats[2].

[2]Source: http://ctb.ku.edu/en/table-of-contents/assessment/assessing-community-needs-and-resources/swot-analysis/main

Functional context and constraints By analyzing the functional context and constraints of a software platform, the research is limited to only two factors instead of four in the SWOT analysis. By describing the functional context of a software platform, new opportunities are explicated, which could enable the mapping of software functionality. An example of a functional contextual parameter of a smartphone device (mobile software platform), is that it is common that the device is always carried by the owner. Since most people own a smartphone, the device and its data are considered to be personal, and devices are not exchanged or shared with other people, which adds contextual information that one can assume that a smartphone is always used by one individual.

Functional constraints of a software platform are concerned with characteristics of a software platform which limit the mapping of functionality, because it would not suit the environment of the new software platform. A functional constraint can originate from both the software platform itself, as from the environment that surrounds the platform. For example, a functional constraint from the platform itself is that the screen size of a smartphone is a lot smaller than the size of a desktop monitor. A functional constraint from the environment is that a user of a smartphone does not always have full attention for the application on the device, as the environment in which it is used is changing and dynamic.

Technical context and constraints As this research project focuses on the functional aspects that play a role in the mapping of software products to another software platform, the precise technical context and constraints of a software platform are of less interest to us. However, it is often the technical capabilities and incapabilities of a software platform that enable or disable certain functionality. Therefore, we state that the technical context and constraints of a software platform need not be explicated when analyzing the software platform. It is possible, however, that a certain aspect of the functional context or constraints is inherent to the underlying technical details of the platform.

4.5.3 Activity and concept table

Table 4.5 gives a summary of the activities within the main activity *Scenario creation* of the Software Functionality Evolution Method.

Main activity	Sub activity	Description
Scenario creation	Define personas	Based on the actual users of the software product, PERSONAS are defined to represent them. A PERSONA can be both an actual as a potential user of the software product, which means that a PERSONA can also represent a user group from a new market segment which is about to be explored. The exploration of PERSONAS is, however, limited by the *Scope* and *Goal* of the PROJECT DEFINITION. In Section 4.5.1, different properties of a PERSONA are explored, which help in the instantiation of the concept.
	Define software platforms	After the PERSONAS in the project have been identified, an analysis of new SOFTWARE PLATFORMS can be made. As with the PERSONAS, the SOFTWARE PLATFORMS align with the PROJECT DEFINITION's *Scope* and *Goal*. Section 4.5.2 explains more about the analysis of SOFTWARE PLATFORMS from a functional perspective.
	Create scenarios	By combining use cases of PERSONAS on the defined SOFTWARE PLATFORMS, SCENARIOS are defined. It is possible that a PERSONA is not present on a SOFTWARE PLATFORM, which means this combination does not produce a SCENARIO.

Table 4.5: Activity table of the SFEM,
main activity *Scenario creation*

Table 4.6 gives a summary of the concepts that are instantiated by the activities within the main activity *Scenario creation* of the Software Functionality Evolution Method.

Concept	Description
PERSONA	PERSONAS are defined as representations of the actual users of a system, defined by the goals they aim to accomplish. Personas are hypothetical archetypes of actual users (Cooper, 1999). A PERSONA appears in at least one SCENARIO, which maps it to at least one SOFTWARE PLATFORM. More information about the defining of a PERSONA can be found in Section 4.5.1.
SOFTWARE PLATFORM	A platform is the combination of an operating system and hardware that makes up the operating environment in which a program runs (ISO/IEC 26513). Thus, a SOFTWARE PLATFORM defines the environment in which a software product is designed to operate. In Section 4.5.2, more information about the description of a SOFTWARE PLATFORM is given.
SCENARIO	The combination of possible and relevant appearances of PERSONAS on SOFTWARE PLATFORMS creates an instantiation of the concept SCENARIO. A SCENARIO is used to map FUNCTIONALITY by MAPPINGS and their *Priorities*.

Table 4.6: Concept table of the SFEM,
main activity *Scenario creation*

4.6 Functionality mapping

The mapping of functionality concerns the application of functionality in a given scenario, thus in the combination of a persona with a software platform. If no instantiation of the concept MAPPING is created, this means the functionality is not relevant for the scenario.

The activities concerning the mapping of functionality on instantiations of the concept SCENARIO lay at the heart of the template method. The activities *Review mapping candidates* and *Prioritize mapping candidates* are performed in parallel in a group session. The preceding activity, *Identify mapping candidates*, prepares the work in the group session by excluding irrelevant mappings, which increases the efficiency of the group session.

During the instantiation of the template method, it is possible that no priority is yet assigned to the mapping. A mapping without priority implies that the mapping is considered a candidate, yet the mapping and its priority are open to further discussion in the group session activity of the template method.

The minimum and maximum value of the priority of a mapping are not strict, and can be different for each template method instantiation, depending on the complexity of the project at hand. For instance, for a simple project, one might use a three-point scale with priority groups, while for a more complex project, one might use a formula to calculate the relative priority on a decimal scale from 0 to 1.

4.6.1 Prioritization techniques

In the field of Software Product Management, prioritization techniques are used in order to assist a software product manager in selecting the right set of requirements for a planned release. For software products, requirements management is driven by the available resources and time, in contrast to tailor-made software, where the amount of resources and time needed are driven by the total amount of requirements. In other words, a software product manager must select requirements based on the available resources and time at hand. Therefore, prioritization techniques assist in selecting requirements for a planned release by filtering only those requirements that are important to the software product and its stakeholders, and can be completed within the limited set of resources and time.

We introduce a set of requirements prioritization techniques that are known in the field of Software Product Management in the sections below. However, the list of techniques is not complete, and always leave room for interpretation or customization to the project at hand. This means that the parameters of a technique can be adjusted, or extended, with the best outcome of the technique in mind.

We acknowledge that there are more (advanced) prioritization approaches, such as the Analytic Hierarchy Process (Karlsson and Ryan, 1997) or the Integer Linear Programming approach (Ruhe and Saliu, 2005; Van den Akker et al., 2008; Carlshamre, 2002). We believe these techniques do not suit the audience of this research - the software product manager - and the purpose of this research - analysis of a software product's functionality on a functional level - and are therefore omitted from this research.

Supported by the research by Berander and Andrews (2005) and Racheva et al. (2008), we give an explanation of different prioritization techniques that are applicable to requirements engineering for software products. Besides the explanation of the technique, we try to identify the strengths and weaknesses of each approach, which helps in selecting the right approach for the project at hand.

A comparison of the characteristics of the techniques is given in Table 4.7, inspired by the research by Berander and Andrews (2005) and Racheva et al. (2008). Prioritization techniques are characterized by *scale*, *granularity* and *sophistication*. The following subsections elaborate on the techniques themselves.

The ***scale*** for the priority values of the technique can be depicted as *nominal*, *ordinal*, *interval* or *ratio* (Stevens, 1946). A *nominal* variable, also known as a *categorical* variable, is for categories that are mutual exclusive, yet no order is depicted among the categories. An *ordinal* variable does depict an order among the categories, however the distance between categories does not matter. For *interval* variables, both the order and the distance between categories is of importance. However, interval variables do not have a clear definition of 0.0, while a *ratio* variable is the same, yet does have a clear definition of a "none" value.

The value in the ***granularity*** column defines the depth of the technique, the extend to which is goes into the details of the requirements. Three values are used, *fine*, *medium* and *coarse*. A *fine* granularity means that the technique requires in-depth knowledge about the requirement, before a priority can be assigned. *Coarse* depicts a very high-level definition of the requirement before a priority can be assigned, and *medium* represents any value in between.

The ***sophistication*** represents the complexity of the technique itself. For instance, it could represent how hard it would be to explain the technique to a group of 25 people, and make sure everyone has the same understanding of the technique and values. Three values are used, *simple*, *medium* and *complex*, of which the second is not applied to any of the discussed techniques.

The explanations for the values in Table 4.7 are given in each last paragraph of the prioritization techniques, discussed below.

Technique	Scale	Granularity	Sophistication
Binary priority list	Ordinal	Medium	Complex
Priority groups	Ordinal	Coarse	Simple
MoSCoW	Ordinal	Coarse	Simple
Requirements triage	Ordinal	Coarse	Simple
Cumulative voting	Ratio	Fine	Complex
Ranking	Ordinal	Medium	Simple
Top ten	Ordinal	Coarse	Simple
Wieger's matrix	Ratio	Fine	Complex

Table 4.7: Comparison of prioritization techniques

Binary priority list The Binary Priority List (BPL, or Binary Search List or Binary Search Tree) is described and researched by Bebensee et al. (2010). The binary search is a popular algorithm to sort and search information (Knuth, 2005), but it can also be used to prioritize requirements (Karlsson et al., 1998). The algorithm aims to create a descending tree of requirements in order of priority, with the most important requirement on the top part of the tree.

The technique includes six steps, of which steps 2, 3 and 4 are repeated for all requirements (Racheva et al., 2008; Karlsson et al., 1998; Ahl, 2005):

1. Gather all requirements and give them a unique identifier
2. Take one random requirement, and use it as the root requirement
3. Take another random requirement and compare it with the root requirement in terms of priority
4. If the new requirement's priority is lower than that of the root requirements, compare it to the requirements below the root, and so forth. If its priority is higher than that of the root requirements, compare it to the requirements above the root, and so forth. Eventually, the process stops when the new requirement has found a place as a sub-requirement of a root requirement which has not yet received its sub-requirement
5. Steps 2, 3 and 4 are repeated for all requirements
6. Eventually, the tree is traversed from top to bottom to get the actual order of prioritized requirements

The image in Figure 4.3 exemplifies a possible outcome of the algorithm. A completed Binary Priority List has to be read from top to bottom, independent of the horizontal position of the elements in the diagram. Given the image in Figure 4.3, the prioritization of the requirements in this example would be as follows: **D, B, E, A, F, C, G**.

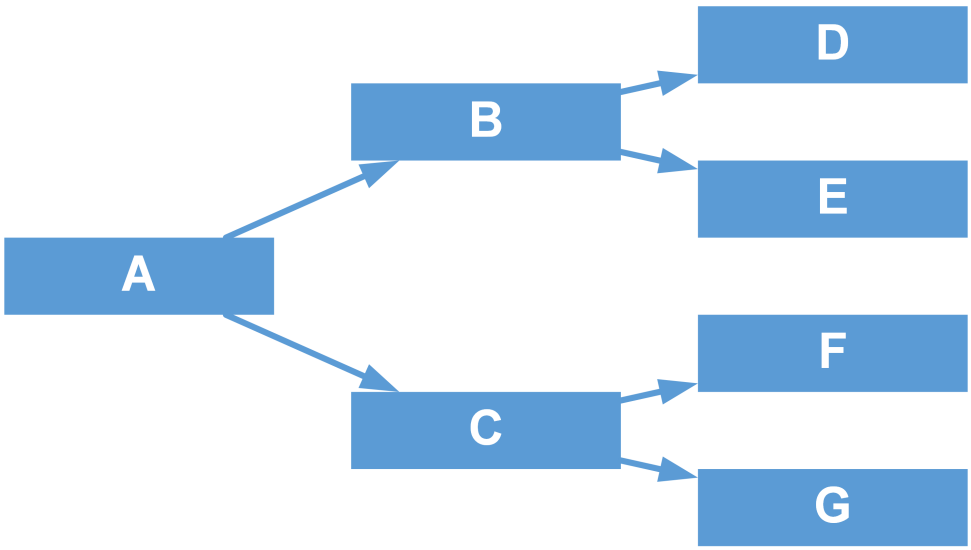


Figure 4.3: Example of the Binary Priority List

This requirements prioritization technique's **scale** is *ordinal*, because the requirements are served in an ordered fashion, yet the distance between two prioritized, subsequent requirements does not imply extra or less importance. The **granularity** is *medium*, as requirements are compared with each other over and over again, giving more insight into the importance of a requirement as the project continues. The **sophistication** is *complex*, because the resulting tree is hard to compose and keep track of, and grows in complexity as more requirements are added and prioritized.

Priority groups The priority groups technique, also known as numerical assignment, is a relatively simple prioritization technique. It has been suggested in RFC 2119 and IEEE Standard 830-1998. The involved stakeholders assign each requirement to a priority group, which is labeled with an absolute term (critical, standard, optional), instead of a relative term which leaves room for interpretation (high, medium, low) (Wiegers, 1999b). According to Leffingwell and Widrig (2003) and Sommerville and Sawyer (1997), a total of three priority groups is common.

As reported by Berander and Andrews (2005), the technique may result in a long list of "critical" requirements. Reported by Berander (2004) and Wiegers (1999b), it is common that *85 percent* is classified as critical, *10 percent* as standard, and *5 percent* as optional. Given that the majority is assigned to the highest priority group, the results of the technique often lack details and depth. Also, the technique does not take the available resources in account, which might result in a list of critical requirements that are still not suitable for implementation in requirements engineering for software products.

Two variations of the priority groups technique are *MoSCoW* and *Requirements triage*, described in the following two sections. Although the techniques are distinct in their approach, the essence remains the same as the priority groups technique.

This requirements prioritization technique's **scale** is *ordinal*, because there is an order among priority groups, yet there is no distinct distance between priority groups. The **granularity** is *coarse*, because no specific attributes from the requirements are required in order to be able to assign a priority group. The **sophistication** is *simple*, because the number of priority groups is often limited to three, and the definition of a category is therefore easy to establish.

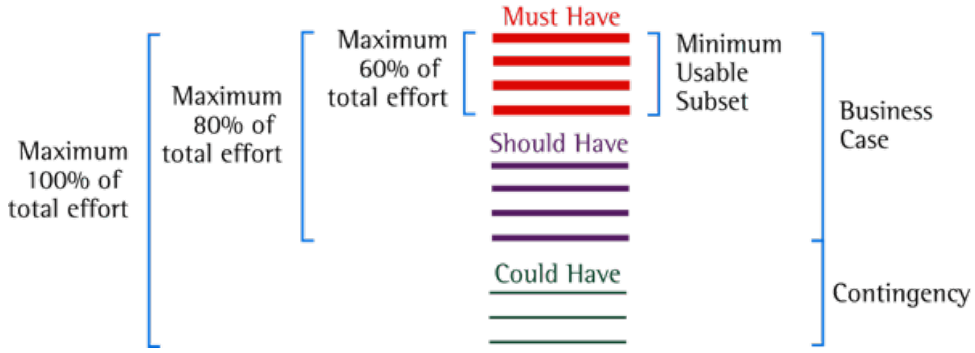


Figure 4.4: MoSCoW prioritization technique (DSDM Consortium, 2008)

MoSCoW The MoSCoW prioritization technique (DSDM Consortium, 2008) is a common classification which can be applied in requirements engineering. The capital letters of the abbreviation MoSCoW represent the following values (DSDM Consortium, 2008):

- **Must have**
  This requirement belongs in the Minimum Usable Subset (MUS), and is of great importance to the success of the project.
- **Should have**
  The requirement is important, but not of vital importance to the project's success.
- **Could have**
  If this requirement were not to be implemented, there is less impact on the project's success.
- **Won't have**
  Any requirement that is excluded from the project, is listed on a Prioritized Requirements List for later review.

The quantification of the prioritization outcome, as described by the DSDM Consortium, is visualized in Figure 4.4. The Figure describes what parts of the prioritized requirements contribute to the total effort of a project, and what part of the total effort belongs to the actual Business Case, and what is left for contingency.

This requirements prioritization technique's **scale** is *ordinal*, because there is an order among the MoSCoW labels, yet there is no distinct distance between any of the groups. The **granularity** is *coarse*, because no specific attributes from the requirements are required in order to be able to assign a label. The **sophistication** is *simple*, because the number of priority groups is limited to four, making it easier to achieve a common understanding of the technique and labels.

Requirements triage Prioritization by means of triage (from the French verb *trier*, meaning to select, sort, order or filter) is a technique that originates from the medical world, and applies very well to the initial prioritization of software requirements and defects. Requirements triage is defined by Davis (2003) as "the process of determining which requirements a product should satisfy given the time and resources available". The original technique is designed to quickly sort patients who seek immediate care from those who can, simply put, wait and be treated later. Many classifications and labels exist, though most of them seem to result in (a variant of) the following classification:

- **High** Those with immediate need for treatment
- **Medium** Those with an urgent need for treatment, yet not critical
- **Low** Those with a need for treatment, yet can wait before those of higher priorities have been treated
- **None** Those with little chance of success after treatment, and can therefore better be omitted in order to treat those with more chance of success

An interesting deviation from the triage application in the medical field, is that for software faults, the *None* category would imply that a software error is so severe, that it would not make any more sense to treat it. Of course, this does not apply to software faults, as those would be even more critical than those in the *High* category. Therefore, we suggest that the *None* category in the field of software engineering implies that the requirement or fault stands no chance to be implemented, as it adds no significant value for the software's users.

This requirements prioritization technique's **scale** is *ordinal*, because there is an order among classifications, yet there is no distinct distance between labels. The **granularity** is *coarse*, because one can remain relatively abstract about a requirement's details before assigning a priority. The **sophistication** is *simple*, because in essence, only four labels are known in the original classification.

Cumulative voting Cumulative voting, also known as the 100 dollar test, is a technique in which the stakeholders are given an imaginary set of 100 units (such as money, hours, or FTEs), which the stakeholders need to distribute between the requirements Leffingwell and Widrig (2003). When the prioritization has completed, the requirements are ranked in a descending order of assigned units.

The technique has some weaknesses, for which the research by Berander and Andrews (2005) summarizes some solutions. One noteworthy downside to the technique, as exemplified in the case study by Regnell et al. (2001), is when the number of requirements increases it becomes more difficult to assign the right number of units to a requirement. A solution to this problem was also proposed in the case study, which involved making more units available per stakeholder. This solution was well received by the stakeholders of the concerning research.

This requirements prioritization technique's **scale** is *ratio*, because requirements are ordered by their score assigned, a zero-value is established, and something can be said about the distance between requirements, based on their assigned score. The **granularity** is *fine*, because a specific value has to be assigned to a requirement as score, and trade-offs need to be made between requirements in order to be able to correctly allocate the available "dollars" to the requirements. The **sophistication** is *complex*, because in order to get to representative values, one must be certain that a common understanding of the project and requirements has been established, before dividing the scores.

Ranking The ranking technique is a relatively simple sorting method which is best suitable for projects with only one stakeholder. The idea is to make a list of all requirements, and sorting them from most important requirement on top, and least important at the bottom.

As noted by Berander and Andrews (2005), the outcome of this technique does not reveal the actual distance between two requirements. Taken for instance the cumulative voting technique, if the first requirement receives $300 and the second $243, we can state there is a distance of $57 between both requirements. With ranking we can only say that requirement 2 is less important than requirement 1, however we can not state how less important it is.

Also noted by Berander and Andrews (2005), when dealing with multiple stakeholders, the technique is less suitable. One could take the mean priority in the order of each requirement, but this could lead to ties in the sorting, which is something the technique tries to avoid in its essence.

This requirements prioritization technique's **scale** is *ordinal*, because by ranking, the requirements are ordered, yet the distance between requirements is of no particular interest or meaning. The **granularity** is *medium*, because requirements need to be compared with each other to be given a priority. The **sophistication** is *simple*, because the concept of ranking is not particularly difficult, by itself.

Top ten When relatively many stakeholders and requirements are involved, applying the top ten technique may offer a suitable solution. Each stakeholder creates a top ten of requirements, without ordering the selected requirements. The outcome of the accumulation of assigned "points" to the requirements, gives a reasonable overview of the requirements that are preferred by the stakeholders.

This requirements prioritization technique's **scale** is *ordinal*, because given the individual top tens and the thus assigned points (from 1 to 10), do not say anything about the precise distance between prioritized and included requirements. Also after accumulating the points per requirement, nothing can be said about the relative distance, because of the lack of distance in the originating scores. The **granularity** is *course*, because it is not possible to go into too much depth for requirements, since only ten requirements can be selected in the end. The **sophistication** is *simple*, because one only needs to select those requirements that are of utmost importance to him/her, and other requirements can be neglected.

Wiegers' prioritization model The requirements prioritization technique by Wiegers (1999a) is a method which involves an extended prioritization of requirements. Instead of applying a single number as a priority to a requirement, it incorporates an estimate of the relative *benefit*, *penalty*, *cost* and *risk* of implementing or omitting a requirement. The *benefit* and *penalty* are often assessed by key customer representatives, and the *cost* and *risk* by development representatives. The relative values of the *benefit* and *penalty* are added up in order to calculate the total *value* of the requirement.

The following approach is proposed by Wiegers (1999a):

1. List all requirements that you want to prioritize
2. Estimate the relative benefit for each requirement on a scale of 1 to 9
3. Estimate the relative penalty for each requirement on a scale of 1 to 9
4. Calculate the total value as relative penalty + relative value
5. Estimate the relative cost for each requirement on a scale of 1 to 9
6. Estimate the relative risk for each requirement on a scale of 1 to 9
7. Calculate the priority and sort the requirements list, ordered descending by the calculated priority

Feature	Relative benefit	Relative penalty	Total value	Value %	Relative cost	Cost %	Relative risk	Risk %	Priority
Query status of a vendor order	5	3	8	8,0%	2	4,8%	1	3,0%	1,027
Generate a Chemical Stockroom inventory report	9	7	16	16,0%	5	11,9%	3	9,1%	0,762
See history of a specific chemical container	5	5	10	10,0%	3	7,1%	2	6,1%	0,757
Maintain a list of hazardous chemicals	4	9	13	13,0%	4	9,5%	4	12,1%	0,601
Print a chemical safety datasheet	2	1	3	3,0%	1	2,4%	1	3,0%	0,554
Modify a pending chemical request	4	3	7	7,0%	3	7,1%	2	6,1%	0,530
Check training database for hazardous chemical training record	3	4	7	7,0%	4	9,5%	2	6,1%	0,449
Generate an individual laboratory inventory report	6	2	8	8,0%	4	9,5%	3	9,1%	0,430
Search vendor catalogs for a specific chemical	9	8	17	17,0%	7	16,7%	8	24,2%	0,416
Import chemical structures from structure drawing tools	7	4	11	11,0%	9	21,4%	7	21,2%	0,258
Totaal	**54**	**46**	**100**	**100,0%**	**42**	**100,0%**	**33**	**100,0%**	

Figure 4.5: Example of the Wiegers' prioritization matrix

The relative variables can be weighed extra or less by assigning a relative weight to the benefit, penalty, cost or risk. This may be applicable to cases where resources are limited or abundant. In Figure 4.5, an example of the Wieger's prioritization matrix is presented.

This requirements prioritization technique's **scale** is *ratio*, because the priority that results from the calculation says something about the distance between requirements, an order can be established, and a zero-point is defined. The **granularity** is *fine*, because the formula includes the four aspects benefit, penalty, cost and risk, of which an assessment needs in-depth knowledge about the requirements. The **sophistication** is *complex*, because the priority is dependent on a formula, different relative weights can be established for the four calculation aspects, and each aspect requires a common understanding of what the aspect means for a requirement.

4.6.2 Design rationale

The concept of a design rationale plays an important role in this research project and the template method deliverable. Since the decision making of mapping software functionality to next-generation software platforms is often based on subjective and tacit knowledge, it is hard for decision makers to communicate and rationalize the outcome of their process. Therefore, we emphasize the important of explication of a design rationale. We use the definition of a design rationale from Lee (1997): "the reasons behind a design decision, the justification for it, the other alternatives considered, the trade offs evaluated, and the argumentation that led to the decision".

A design rationale thus helps to communicate the reasoning behind decision making, and can be seen as the basis for documentation of the project. Based on the complexity, a template for the structure of the explication can be designed, which helps to quickly cover all required aspects of the design rationale, such as those from the above mentioned definition by Lee (1997). However, we find it important to state that capturing a design rationale is purely a means to an end, and should therefore not be seen as a required concept. In the method we are designing, we make a design rationale optional, which prevents too much overhead for design decisions that are too simple or logical.

4.6.3 Activity and concept table

Table 4.8 gives a summary of the activities within the main activity *Functionality mapping* of the Software Functionality Evolution Method.

Main activity	Sub activity	Description
Functionality mapping	Identify mapping candidates	By iterating over the instantiations of FUNCTIONALITY for each SCENARIO, an initial mapping is made which defines whether or not it would make sense to provision the FUNCTIONALITY for a given SCENARIO. If so, an instantiation of the concept MAPPING is made, with an empty *Priority* property. If the FUNCTIONALITY should not be provisioned for the given SCENARIO, no MAPPING is instantiated.
	Review mapping candidates	The reviewing of MAPPING candidates is performed in a group session with selected and relevant STAKEHOLDERS. The initial MAPPINGS are discussed and MAPPINGS can be added or removed. No *Priority* is yet assigned to the MAPPING during this activity.
	Prioritize mapping candidates	Parallel with the activity *Review mapping candidates*, the MAPPING candidates are reviewed by assigning a *Priority*. The activity is performed in a group session with selected and relevant STAKEHOLDERS. The complexity of this activity depends on the complexity of the project instantiation, the complexity of the software product and the depth of the analysis of FUNCTIONALITY. If the decision of a MAPPING is complex or not sufficiently obvious, the decision can be captured in a DESIGN RATIONALE. Different prioritization techniques, which can help in the calculation of the *Priority*, are described in Section 4.6.1.

Table 4.8: Activity table of the SFEM, main activity *Functionality mapping*

Table 4.9 gives a summary of the concepts that are instantiated by the activities within the main activity *Functionality mapping* of the Software Functionality Evolution Method.

Concept	Description
MAPPING	A MAPPING is an assigned correspondence between two things that is represented as a set of ordered pairs (IEEE Std. 1320.2-1998). The MAPPING concept plays a central role in the REPORT of the results of a template method instantiation. A MAPPING is created by the combination of a SCENARIO with FUNCTIONALITY. If no mapping occurs at the SCENARIO, meaning that a certain FUNCTIONALITY is excluded from a SCENARIO, the MAPPING does not exist. The *Priority* of the MAPPING is determined based on the importance of a set of FUNCTIONALITY for a given SCENARIO. If a MAPPING has no *Priority* assigned, it is considered to be a candidate. The decision and rationale behind a MAPPING can be captured in a DESIGN RATIONALE.
DESIGN RATIONALE	A DESIGN RATIONALE is defined as information capturing the reasoning of the designer that led to the system as designed, including design options, trade-offs considered, decisions made, and the justifications of those decisions (IEEE Std. 1016-2009). It presents the arguments behind a MAPPING and its *Priority*. If the MAPPING changes due to an iteration after the *Results reporting* activities, another DESIGN RATIONALE can be assigned to the same MAPPING. More information about the concept DESIGN RATIONALE can be found in Section 4.6.2.

Table 4.9: Concept table of the SFEM, main activity *Functionality mapping*

4.7 Results reporting

Finalizing an instantiation of the template method, the activities in the main activity *Results reporting* communicate the results of the instantiation to selected and relevant stakeholders. The reporting of a project is important to a successful completion, since the stakeholders of the project need to be properly informed and agree to the outcome of the instantiation. Different stakeholders in the project can have different objectives, and therefore need to make an assessment of the reported outcome, to determine whether it suits their needs.

If the stakeholders to whom a report has been handed, are not able to agree on the outcome of the instantiation, it may be decided to return to the *Functionality mapping* activities. If so, it is advisable to include those stakeholders in the *Functionality mapping* activities, as well. After another iteration of mapping has been performed, the results of the project can be reported again, after which another assessment of consensus needs to be made.

It is possible that the activities in the main activity are repeated, regardless of an agreement on consensus or not, for instance when it has become clear that another selection of stakeholders needs to be informed, also requiring a different design of the concept REPORT. However, we have not modeled this into the template method, as it is an exception which is not certain to occur.

The outcome of a template method instantiation does not create a product roadmap. A product roadmap is highly dependent on the available resources and required resources per requirement, which is something that was consciously omitted from the template method. If resources were to be incorporated into the mapping and prioritization activities of the template method, the process would have become too devious, significantly increasing the time required for a template method instantiation and reducing its efficiency. It is possible that the report of an instantiation is used as input for the product roadmap, as was also discussed in Section 1.4.

4.7.1 Mapping matrix

A suitable way to present the report of a template method instantiation is by what we call a "mapping matrix". Different variations of mapping matrices are presented in literature, of which an excellent example is the domain mapping matrix (Danilovic and Sandkull, 2005). The research helps understanding the interdependencies between items, components, organizations, teams, or people, where complexity arises from the relationships and dependencies among items.

In terms of this research project, a mapping matrix serves a similar cause. It helps to understand the relationships between sets of functionality and scenarios of personas acting on a software platform. The mapping entity of the template method, optionally with a corresponding design rationale, defines the existence of a relationship between an entity of functionality and an entity of a scenario.

Simply put, a mapping matrix is a matrix with functionality the vertical axis, and scenarios on the horizontal axis. The mappings between functionality and scenarios are represented in the cells of the matrix, by means of their assigned priority. Different variations and levels of detail can be added or omitted, dependent on the stakeholders for whom the report is designed. The functionality on the vertical axis can be further elaborated by showing the related entity in another column, as well as related groupings from the domain ontology.

4.7.2 Categorization

An important side note about the results of an instantiation of the template method is that it does not take into account the relationships and dependencies between functionality. However, when the report of the outcome of the instantiation is designed, categorization can be applied to the functionality, based on their mappings. Different categorizations are presented below. By means of categorization, the designed report can be tuned more towards strategic planning, which suits the desired information for a product roadmap better.

Per component By categorizing requirements per architectural component, a software product manager is assisted in the creation of a product roadmap because functional dependencies or relationships between functionality are identified. Especially when seen from a top-down perspective, when deciding about an entity having to be mapped in a given scenario, grouping by component helps to make decisions about a large set of mappings at the same time.

Per product If a software suite exists of different types of products (such as ERP, CRM, and HRM), deciding about all entities and functionality within a given product can help speeding up the decision making process when creating a product roadmap. This categorization also helps in the analysis of the impact of a mapping decision, as it reveals which products are impacted by a decision.

Per industry If a software developing organization wishes to categorize its product roadmap per industry, it helps in identifying strategic focus areas which provide a competitive advantage for the software product. Different roadmaps thus have a different theme, based on the industry. This can either be an industry in which the organization is already active, or an industry to which they wish to expand.

Per release A categorization per release can help an organization in making both a short and long term planning for their product releases, and which requirements will be implemented in those releases. This categorization is practically already the creation of a product roadmap, although the product manager could decide not to take available and required resources into account, yet.

4.7.3 Activity and concept table

Table 4.10 gives a summary of the activities within the main activity *Results reporting* of the Software Functionality Evolution Method.

Main activity	Sub activity	Description
Results reporting	Select stakeholders	Before a REPORT can be created, a set of STAKEHOLDERS has to be selected, for whom to report to. The interests of the STAKEHOLDERS determine the level of detail in the REPORT.
	Create report	Tuned to the interests of the selected STAKEHOLDERS, a REPORT of the project instantiation's outcome is created. In Section 4.7, more information about the concept REPORT and categorizations is given.
	Report to stakeholders	The created REPORT is presented to the selected STAKEHOLDERS, and a discussion is held to determine whether there is consensus about the REPORT and results being satisfying to all STAKEHOLDERS. If the STAKEHOLDERS can not reach consensus about the results in the REPORT, a new group session is initiated to review the MAPPINGS. In the case where no new iteration over the MAPPINGS has to be made, the project instantiation comes to an end.

Table 4.10: Activity table of the SFEM, main activity *Results reporting*

Table 4.11 gives a summary of the concepts that are instantiated by the activities within the main activity *Results reporting* of the Software Functionality Evolution Method.

Concept	Description
REPORT	A REPORT is an information item that describes the results of activities such as investigations, observations, assessments, or tests (ISO/IEC/IEEE 15289). The results of a template method instantiation are communicated to selected STAKEHOLDERS by a REPORT, which is designed to suit the STAKEHOLDER's interests. As the needs of the STAKEHOLDER, and thus the details of the REPORT, highly depend on the situation at hand, we define the REPORT as a closed complex concept. A REPORT incorporates at least one or more instantiations of the concepts PROJECT PLAN, FUNCTIONALITY, SCENARIO and MAPPING. Section 4.7 describes possible formats of a REPORT.

Table 4.11: Concept table of the SFEM, main activity *Results reporting*

Chapter 5

Template method instantiations

In two case studies, the designed template method was instantiated, creating a backlog of the activities conducted and concepts produced. This chapter describes how the method has evolved by means of creating method increments and implementing these in the design of the Software Functionality Evolution Method (SFEM). Section 5.1 proposes a categorization for the reflection on method increments, followed by the increments that were designed between versions of the SFEM. In Section 5.2, we give a summary of the case studies that were performed, of which a complete backlog is available in the appendices.

5.1 Template method increments

This section describes how the Software Functionality Evolution Method (SFEM) has evolved during the execution of the research project, by means of template method instantiation. The instantiation of a template method gives insight into the performance of a template method. The performance is analyzed, which allowed us to identify method increments. These increments serve to improve the SFEM.

As described in Section 2.2.1, we are inspired by the constructivist hermeneutic framework by Van der Schuur (2011) as a framework to reflect on the method increments that we identify. In Figure 5.1, the incremental approach by means of method increments and method improvement is visualized. Both the template method and the template method instantiation are used for the analysis of the method's performance. This results in issues, to be resolved by method increments. The method increments are implemented into the template method to improve it and increase its effectiveness and efficiency. When another iteration of the template method is designed, it is ready to be instantiated again, so that the analysis can be repeated and the method can keep improving.

In earlier work by Van de Weerd et al. (2007), the following elementary increment types have been distinguished:

- *insertion* of a concept, property, relationship, activity node, transition, role
- *modification* of a concept, property, relationship, activity node, transition, role
- *deletion* of a concept, property, relationship, activity node, transition, role

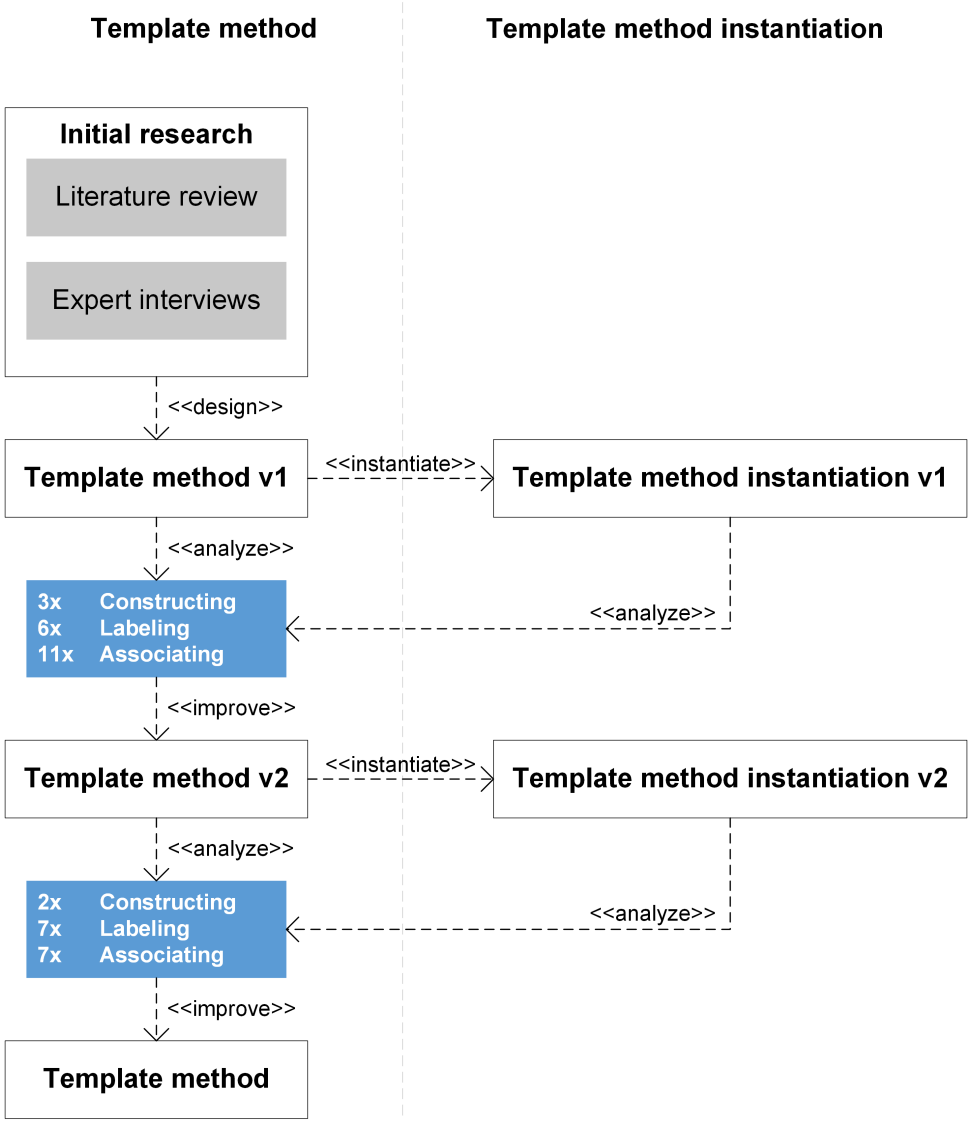


Figure 5.1: Incremental method engineering approach

However, we believe these increment types do not take into account reasoning and motivation for usage, which limits the researcher correctly reflecting on the incremental method engineering. Therefore, we propose a categorization for method increments, which has been established based on analysis of the incremental method engineering conducted during the construction of the Software Functionality Evolution Method, and the method increments resulting from the process. The method increment categorization allows for reflection on the research process, instead of solely naming and applying the method increments. The categorization is presented in Table 5.1.

Category	Description
Constructing	Adding or removing (main) activities, concepts or properties in the diagram. Changing activity or concept types to simple or complex and to open or closed is also considered Constructing. Associations that comes with new (main) activities or concepts are not considered as Associating, but are part of the Constructing.
Labeling	Adding, changing or removing a label of (main) activities, concepts, properties, associations or roles. This prevents confusion or removes any room for interpretation.
Associating	Adding, changing or removing associations between existing (main) activities or concepts. For activities, this includes normal, conditional, unordered and concurrent associations, and changing the direction of the association or the order of the activities. For concepts, this includes normal associations, aggregations and generalizations, and the multiplicity of associations. This also applies to connections between activities and concepts. Creating associations that come with new (main) activities or concepts is considered Constructing.

Table 5.1: Method increment categorization

The incremental method engineering approach has been introduced in Figure 5.1. The figure depicts how the initial research contributed to the first version of the template method, and how analysis of the template method and instantiation have provided method increments to improve the next version of the template method. Furthermore, the cumulatives of the method increments that resulted from the instantiations have been included presented. These cumulatives are specified in the following subsections, in which the method increments are presented.

5.1.1 Towards the initial version

The following method increments have been captured during the engineering of the initial version of the SFEM. The fact that we have designed method increments even before instantiating the method is something that may require extra explanation. The initial version of the SFEM has been designed based on the initial research, as seen in Figure 5.1, which conducted a literature review and expert interviews.

After the very first version of the SFEM had been designed, it became possible to start reasoning about the method in terms of the Process-Deliverable Diagram (PDD) that was designed as part of them SFEM. Because we had a PDD to go with the method, improvements to the method could now be captured as method increments. This is why we have captured method increments, even before we have instantiated the template method and had a backlog of its performance to reflect on.

Based on the initial research and the application of these method increments, the initial version of the SFEM has been designed, which is visualized as a PDD in Figure 5.2.

Constructing We need an activity to select the correct STAKEHOLDER for which to design a VISUALIZATION. This makes it possible to tune the level of detail in the VISUALIZATION to the selected STAKEHOLDER.

Constructing A STAKEHOLDER should have the property *Role*. This prevents a too shallow description of the concept STAKEHOLDER.

Associating The activities *Review mapping candidates* and *Prioritize mapping candidates* should be concurrent activities, because they can occur simultaneously in the same session, and they are not consecutive.

Labeling The activity *Identify absolute mapping* is ambiguous. It would be better to rename this activity to *Identify mapping candidates*.

5.1.2 Towards the second version

After the first version of the template method had been instantiated in the first case study, the backlog (captured in Appendix D) allowed for analysis of the performance of the template method. Combined with an analysis and reasoning about the design of the template method itself, the following list of method increments has been captured.

Based on the analysis of the initial template method and its performance during the first template method instantiation, the second version of the SFEM has been designed, which is visualized as a PDD in Figure 5.3.

Constructing The activity *Design visualization* is a closed activity. It can be an open activity, but we would have to define the sub-activities that need to be undertaken.

Associating The main activity *Visualization* is not a singular process. It should be incremental, so that multiple VISUALIZATIONS can be designed for different STAKEHOLDERS.

Constructing We need to capture documentation to reside with the VISUALIZATION. Instead of adding an explicit concept, a VISUALIZATION includes the concept DESIGN RATIONALE. This implies that a VISUALIZATION does not necessarily need to be a figure, it can just as well be textual.

Associating A VISUALIZATION must also include the concepts FUNCTIONALITY and ENTITY.

Associating There must be an association between the concepts STAKEHOLDER and VISUALIZATION, because a VISUALIZATION is designed for one or more STAKEHOLDERS.

Constructing In the *Peer review* activity, a VISUALIZATION of the method's output must be included in order to review the performance. Therefore, we suggest merging the *Peer review* activity with the main activity *Visualization*. This implies that after the *Visualization* activity, we must also be able to return to the *Mapping* main activity.

Associating The concepts that are included in a VISUALIZATION may not all have a multiplicity of at least one. For instance, when we design a visual roadmap, the instantiated concepts ENTITY and DATA MODEL are not necessarily included in the deliverable. We must carefully review the multiplicity of associations.

Associating The aggregation of the concept VISUALIZATION adds many lines to the Process-Deliverable Diagram, which makes it harder to read. If we change the color of the lines of aggregations, this would increase the ease of interpretation of the diagram.

Associating You should determine STAKEHOLDERS only after the project's scope has been set, otherwise it is not certain where to find them.

Associating We can't set the project's goal without knowing what the scope actually is we're working in. Therefore, the activity *Set project scope* is implied by the activity *Define migration project*.

Associating The concepts SOFTWARE PLATFORM and PERSONA both appear in at least one SCENARIO, otherwise it would not make sense to define either of the concepts at all.

Associating If we define the PERSONAS before we define the SOFTWARE PLATFORMS, we can select the corresponding SOFTWARE PLATFORMS that the PERSONAS are actually working on. The driving goal of the method is to address the needs of PERSONAS, not that of the SOFTWARE PLATFORMS.

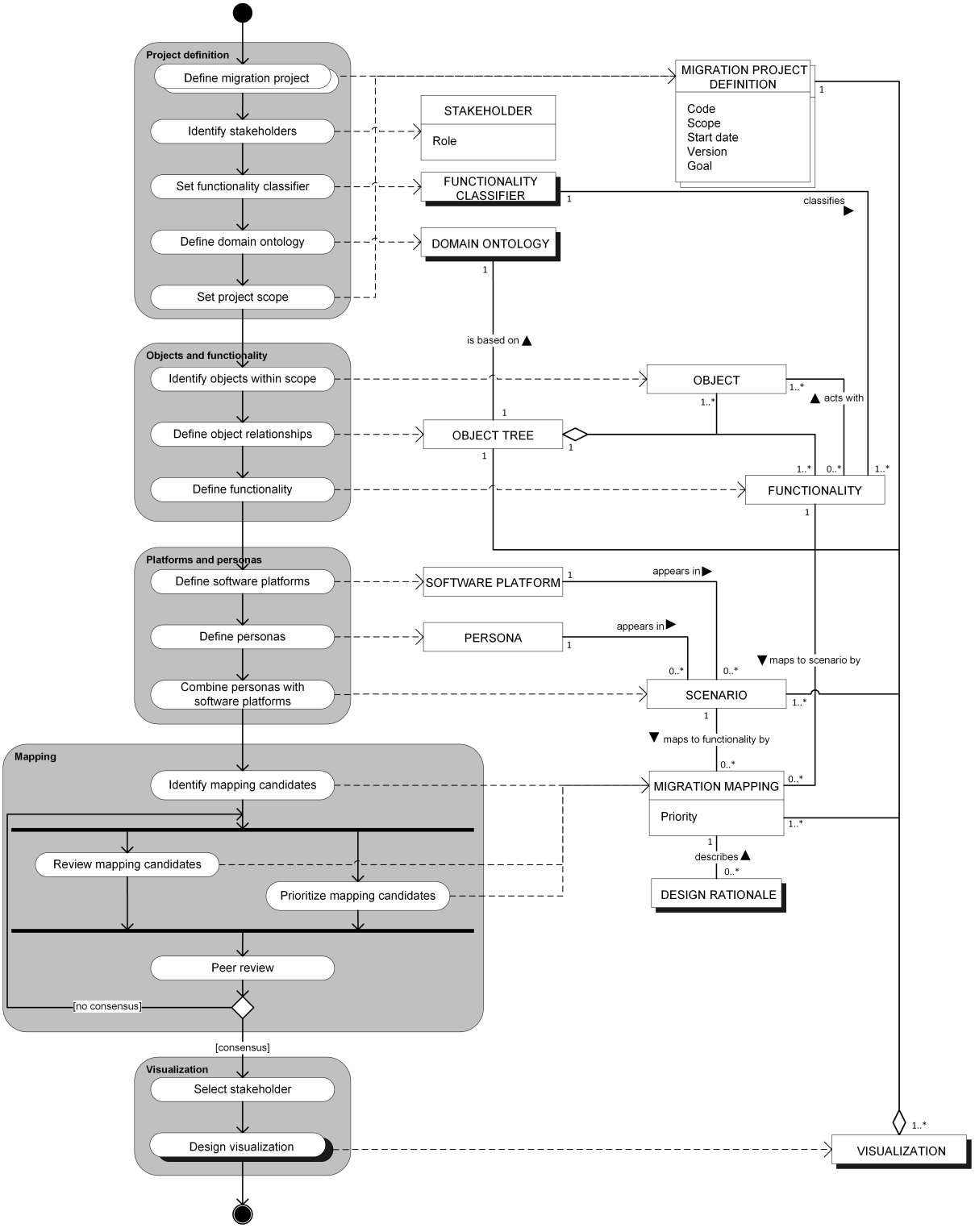


Figure 5.2: Initial version of the Software Functionality Evolution Method

Labeling The main activity *Platforms and personas* should be renamed to *Scenarios*, as the aim of the main activity is to develop SCENARIOS.

Labeling The term for the concept OBJECT is ambiguous. The term ENTITY is more suitable, considering the method domain's jargon.

Labeling The concept OBJECT TREE should be renamed to DATA MODEL, considering the method domain's jargon.

Associating—Labeling The relationship between DOMAIN ONTOLOGY and DATA MODEL should be "DOMAIN ONTOLOGY structures DATA MODEL" or "DOMAIN ONTOLOGY gives structure to DATA MODEL".

Labeling The activity *Define functionality* should be renamed to *Identify functionality*, because the defining of FUNCTIONALITY is something that has already been done in the original design of the software product. Now, we are simply identifying the already defined FUNCTIONALITY.

Constructing The concept FUNCTIONALITY CLASSIFIER does not contribute significantly to the method's performance and deliverables. It would be better to leave it out of the method and describe it in the activity and concept tables. For instance, we could describe it in the description of the activity *Identify functionality*.

Labeling The main activity *Objects and functionality* should be renamed to *Functionality*, as the identification of FUNCTIONALITY is the main focus of the sub activities.

Associating By performing the activity *Identify functionality* before performing the activity *Define entity relationships*, you can use the instantiations of FUNCTIONALITY to identify relationships amongst ENTITIES.

5.1.3 Towards the final version

The final version of the SFEM has been designed based on method increments which have been captured based on analysis of the second version of the SFEM and an analysis of the backlog of the second template method instantiation, represented in Appendix E.

The final version of the template method is visualized as a PDD in Figure 5.4, and further described in Chapter 4.

Associating The association between DATA MODEL and FUNCTIONALITY must be removed. Adding relationships between FUNCTIONALITY next to the relationships between ENTITIES makes a diagram unnecessarily large.

Associating The activities *Define personas* and *Define software platforms* should be parallel activities, since the identification of a SOFTWARE PLATFORM can also complement the identified PERSONAS.

Labeling The activity *Combine personas with software platforms* should be renamed to *Create scenarios*. The way to create scenarios, by combining PERSONAS with SOFTWARE PLATFORMS, will be elaborated in the activity table.

Labeling The concept VISUALIZATION should be renamed to REPORT, as the deliverable is not necessarily a visual representation in form of a diagram of figure, but can also be a textual document or matrix.

Constructing The main activity *Visualization* is not iterative by default. Therefore, we do not return to the activity *Select stakeholders* after reporting to STAKEHOLDERS. However, we can still go back in the method when the STAKEHOLDERS can not reach consensus about the reported deliverable.

Labeling The activity *Design visualization* must be renamed to *Create report*, conform the method increments mentioned earlier in this method revision.

Labeling The main activity *Visualization* must be renamed to *Results reporting*, conform the method increments mentioned earlier in this method revision.

Constructing We need an activity *Report to stakeholders* to complete the main activity *Results reporting*, before we can end the method or return to the main activity *Mapping*.

Labeling We rename the concept MIGRATION MAPPING to MAPPING, since we want to avoid wrong interpretations of its purpose considering the thought that *migration* means movement from A to B.

Labeling The concept MIGRATION PROJECT DEFINITION is renamed to PROJECT PLAN, to conform to the IEEE Standard 610.12-1990 definition. The concept itself remains the same.

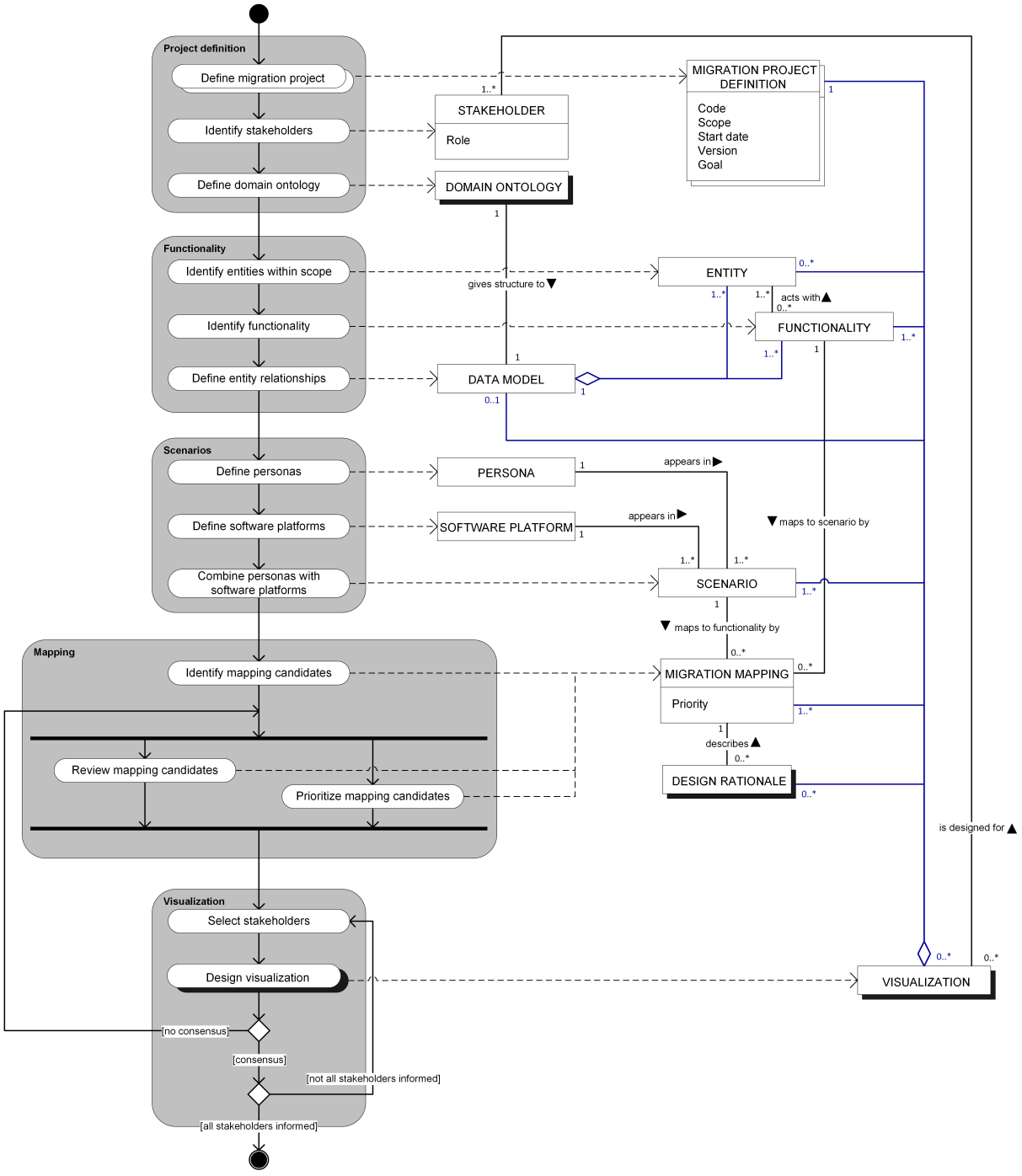


Figure 5.3: Second version of the Software Functionality Evolution Method

Labeling The activity *Define migration project* is renamed to *Write project plan*, to adhere to the change of the MIGRATION PROJECT DEFINITION label.

Labeling The main activity *Functionality* is renamed to *Functionality identification*, to change the label from a passive to an active state.

Labeling The main activity *Scenarios* is renamed to *Scenario creation*, to change the label from a passive to an active state.

Labeling The main activity *Mapping* is renamed to *Functionality mapping*, to change the label from a passive to an active state.

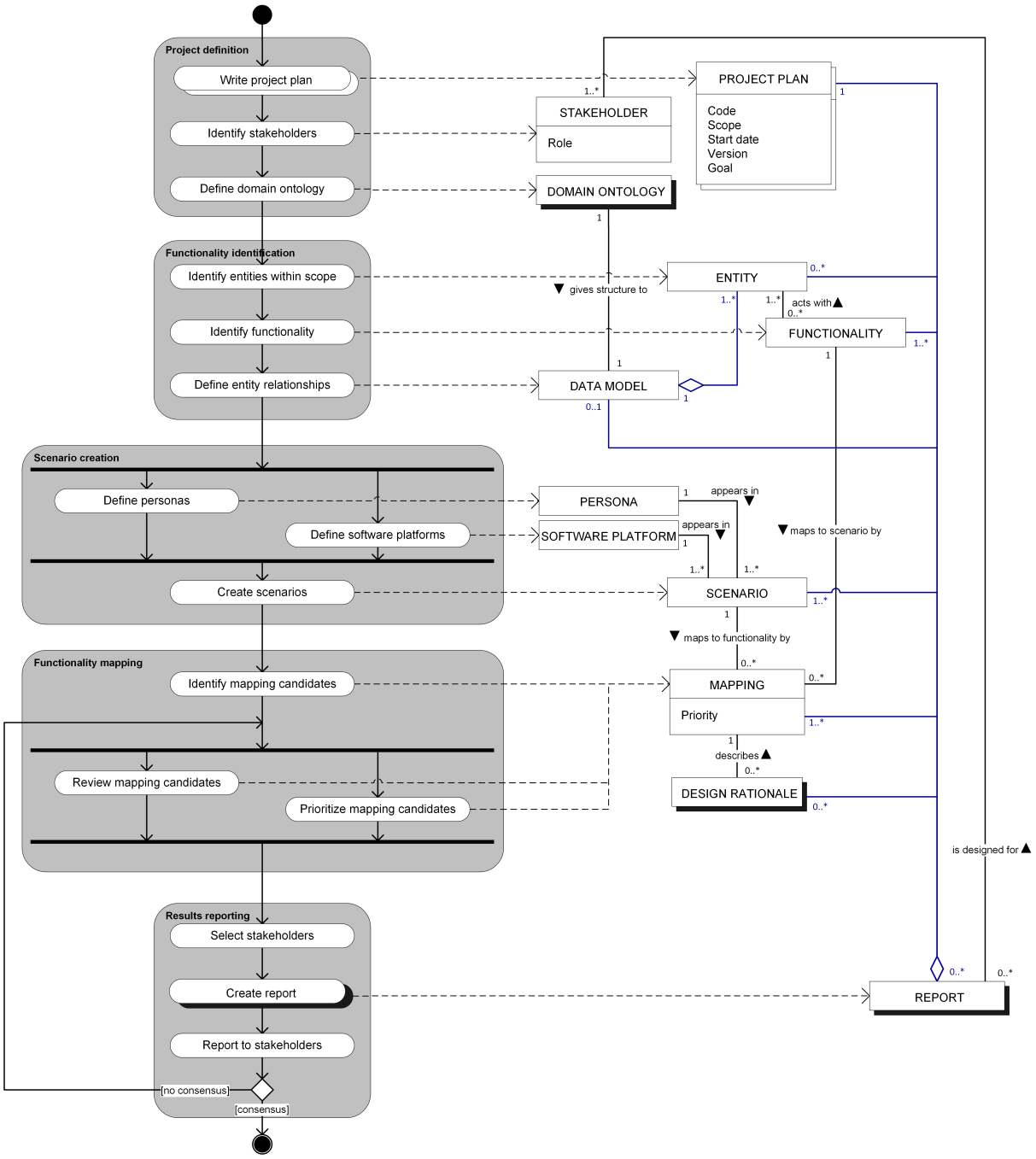


Figure 5.4: Final version of the Software Functionality Evolution Method

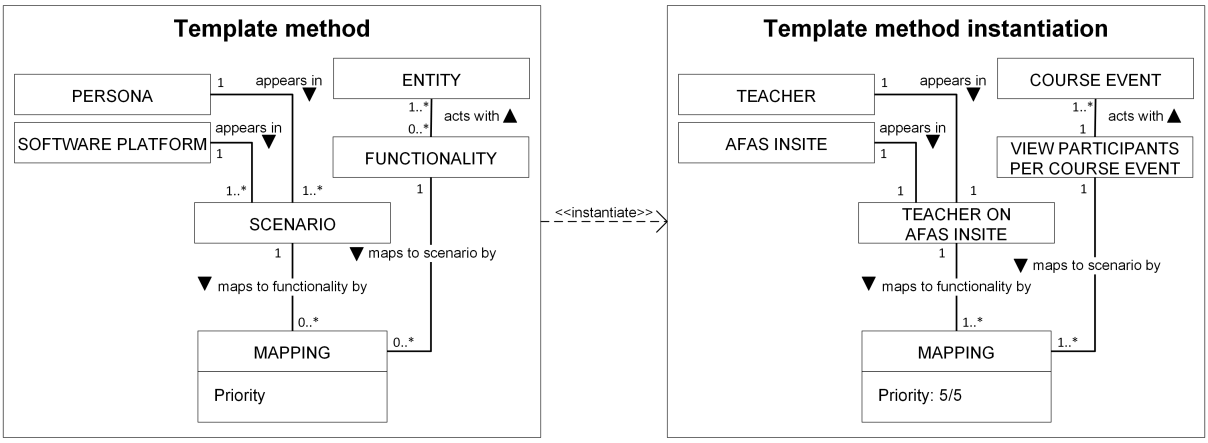


Figure 5.5: Template method instantiation for Course management

5.2 Case study summaries

In the appendices of this thesis, two case studies are reported as being part of the incremental cycle of method engineering to design the template method in this research project. As explained in the research approach (Chapter 2) and supported by the research by Van der Schuur (2011), instantiating a template method enables analysis of its performance and allows to improve the method by designing method increments.

For the sake of visualizing what a template method instantiation looks like, we present a snippet of the template method instantiation in the first case study in Figure 5.5. The visualization shows how the concepts surrounding the concept MAPPING are instantiated. The functionality, VIEW PARTICIPANTS PER COURSE EVENT, is given a priority of five out of a maximum of five points, given the scenario of a TEACHER working on the AFAS INSITE platform. The priority of the MAPPING has been decided upon during the group session, in which it became clear that during a course day, a teacher is particularly interested in the number of participants and their names and organizations, which helps a teacher to prepare for the course event.

5.2.1 Course management in AFAS InSite

This case study[1] aims to perform a mapping of the functionality in the function group *Course management* to the software platform AFAS InSite. AFAS InSite is a web application designed to be used as an intranet for organizations. This means that the functionality in the platform is designed for authorized employees of the organization, and no external entities are supposed to have access to the system. Although we can imagine that functionality from *Course management* may also be of interest to individuals outside an organization, such as external participants or teachers, this case study has restricted itself to the intranet application, due to the particular scope and goal of the case study.

As a matter of fact, a larger part of the functionality related to *Course management* has already been implemented in AFAS InSite. However, in correspondence with the case company, we have decided to initiate our template method instantiations and case studies with this function group in scope, as it is an excellent approach to compare the outcome of the template method with the current state of the software evolution, which is believed not to follow a structured approach as captured by the template method.

[1] Please refer to Appendix D for the complete backlog of this case study.

During the execution of the case study, the researchers developed the Method Stakeholder Classification Matrix (MSCM), which assists in the identification of stakeholders and their roles in the instantiation of a template method. The identified stakeholders are 2 *Participants*, 1 *Informer*, and 2 *Observers*.

After defining the domain ontology, a total of 11 instances of the concept OBJECT were identified directly within the function group in scope. However, since objects do not necessarily have to be inside the function group in order to be relevant to the case study (such as the object DOSSIERITEM), an extended list of 29 objects was included in the case study. Assisted by the functionality classifier CRUD, a total of 135 sets of functionality was identified.

As discussed earlier in this section, the case study was limited to only the software platform AFAS INSITE. Three personas were identified: EMPLOYEE, TEACHER and COURSE MANAGER. Given the single software platform, this also creates three instantiations of the concept SCENARIO.

From the identified functionality, 90 of the 135 sets were never assigned a mapping, excluding them from the software evolution. Given the three scenarios and 135 sets of functionality, a maximum of 405 mappings could have been defined. After the group sessions, 97 mappings and their priorities were defined, which means that 24 percent of the possible mappings ends up as a mapping.

The desired specifications of a visualization were briefly discussed with the stakeholders in the case study. It became clear that a matrix of functionality, scenarios and mappings would be the most suitable way to present the outcome. The columns represent scenarios, rows represent functionality, and the intersections contain the mappings. By sorting the functionality descending by their average mapping priority, the most obvious functionality to put on a product roadmap would be presented at the top of the matrix, and the least important functionality would be at the bottom, or even excluded.

In the first paragraph of this case study summary, we pointed out that the aim of the case study was to compare the outcome of an instantiation with the actual, completed implementation of the function group *Course management* in AFAS InSite, which has been done without use of the template method. After an analysis of the implemented functionality and the results of the template method instantiation, we can state that the results are relatively the same. The functionality of the entity AFBEELDING is already implemented, as well as the functionality which enables the viewing/reading of the primary entities of *Course management*, such as CURSUS, CURSUSSESSIE, DOSSIERITEM and EVENEMENT.

However, if we look closely at the current implementation of functionality, we see that most of the current functionality is centered around read-only operators of entities, and not the actual management of entities. For instance, it is currently not possible to add new participants to a course event, nor can you add a new course event to a course. This differentiates from the results of the case study, where managing operators were not explicitly excluded from the mappings.

From the case study results, another interesting facts caught our attention. In the case study, the functionality to create an instantiation of the entity ORGANISATIE/PERSOON had been given a relatively high average priority of 3.3, yet it has not been implemented into AFAS InSite. This was explained by the product managers by the fact that this functionality has a high complexity, due to the many dependencies from other entities and function groups. This means that if the functionality were to be implemented in AFAS InSite, related entities would need to be available as well, in order for the functionality to be usable.

5.2.2 Fixed assets in AFAS InSite

The second case study of the research project[2] aims to perform a mapping of the functionality in the function group *Fixed assets*, again towards the software platform AFAS InSite. AFAS InSite is a web application designed to be used as an intranet for organizations. This means that the functionality in the platform is designed for authorized employees of the organization, and no external entities are supposed to have access to the system. For the functionality and entities of *Fixed assets*, the limitation of the case study to only the software platform AFAS InSite makes sense, since a fixed asset is an entity which is supposed to be handled inside an organization, and no external entities are expected to work on fixed assets within an organization's administration.

Compared to the first case study, this case study aims to perform a mapping of functionality that has not yet been implemented on the new software platform, instead of trying to compare results of the instantiation with an actual completed implementation. Since none of the product managers said to have given any particular attention to the mapping of functionality in this function group, yet, it is interesting to see how the template method performs in such circumstances.

Again assisted by the Method Stakeholder Classification Matrix, four stakeholders were identified, a *Participant*, a *Participant–Informer*, an *Observer–Informer*, and an *Informer*. The *Observer–Informer* stakeholder played two explicit roles in the case study, which is why we assigned two roles.

Supported by the ontology of entities in the software product, 28 instantiations of the concept ENTITY were identified. 79 sets of functionality were applicable to the entities, of which the identification was supported by the CRUD classifier, the AFAS Knowledge Base and the course *Financieel Procesbeheer*, provided by AFAS Software. In a consult with one of the stakeholders, the relevance of a few entities was discussed, which resulted in the exclusion of two entities and their functionality from the case study.

As the research project had progressed, the founding theory of the template method now allowed for a more structured and complete definition of the concepts PERSONA and SOFTWARE PLATFORM. Therefore, elaborate descriptions of five personas were given. The personas include EMPLOYEE, FACILITY MANAGER, ICT MANAGER, FINANCIAL CONTROLLER and CHIEF FINANCIAL OFFICER. A total of five software platforms was identified, although only the platform AFAS InSite was relevant in this case study.

In the group session with selected stakeholders, the functionality was discussed in order to create instantiations of the concept MAPPING. 79 sets of functionality were discussed, of which 56 sets were assigned a mapping and priority. This implies that 23 sets of functionality were excluded from the software evolution.

Because an instantiation of the VISUALIZATION[3] had already been discussed in the first case study, for this case study the stakeholders was asked whether or not the previous visualization was satisfying. It was pointed out that the visualization was sufficient and suitable for its goal: input for product roadmapping. An important sidenote given was that the matrix should be sorted by the total score of priority, divided by the total amount of scenarios, instead of divided by the amount of mappings for the given functionality.

In Figure 5.6, we present the instantiation of the concept VISUALIZATION for the case study *Fixed assets in AFAS InSite*. As discussed in the previous paragraph, the stakeholders decided to have the requirements prioritized by the final column, which is calculated by summing up the total priority score, and dividing it by the total number of possible scenarios. Given the third row, with a total priority score of *15*, divided by a total of *5* scenarios, the average priority score would be *3.0*.

[2] Please refer to Appendix E for the complete backlog of this case study.

[3] This concept was renamed to REPORT after the instantiation of this case study, because of a formalization of terminology, captured by a method increment.

Functiegroep	Gegeven	CRUD	Functionaliteit	Employee	Facility manager	ICT manager	Financial controller	Chief financial officer	SUM	COUNT	AVG	AVG/5
Algemeen	Afbeelding	R	Afbeelding weergeven	5	5	5	5	5	25	5	5,0	5,0
Vaste activa	Actief	R	Actief weergeven	4	4	4	5	5	22	5	4,4	4,4
Vaste activa	Actief	C	Actief toevoegen		5	5	5		15	3	5,0	3,0
Algemeen	Afbeelding	CD	Afbeelding toevoegen		5	5	5		15	3	5,0	3,0
Algemeen	Afbeelding	D	Afbeelding verwijderen		5	5	5		15	3	5,0	3,0
Organisatie/persoon	Medewerker	R	Activa van een Medewerker raadplegen	5			5	5	15	3	5,0	3,0
Vaste activa	Actief	U	Gekoppeld actief aan actief toevoegen		4	4	5		13	3	4,3	2,6
Vaste activa	Actief	U	Actief bewerken Algemeen		2	5	5		12	3	4,0	2,4
Vaste activa	Factuur	R	Factuur van Actief weergeven		3	3	3	3	12	4	3,0	2,4
Dossier	Dossieritem	C	Dossieritem toevoegen	2	2	2	2	2	10	5	2,0	2,0
Dossier	Dossieritem	C	Bijlage aan dossieritem toevoegen	2	2	2	2	2	10	5	2,0	2,0
Dossier	Dossieritem	R	Dossieritem weergeven	2	2	2	2	2	10	5	2,0	2,0
Dossier	Dossieritem	U	Dossieritem bewerken	2	2	2	2	2	10	5	2,0	2,0
Crediteur	Crediteur	R	Crediteur weergeven				4	4	8	2	4,0	1,6
Vaste activa	Actief	R	Rapportage Activa stamkaart raadplegen				4	2	6	2	3,0	1,2
Vaste activa	Actief	R	Journaalposten van Actief raadplegen				4	2	6	2	3,0	1,2
Vaste activa	Commerciële aanschaffingsstaat	R	(Commerciële) Aanschaffingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Commerciële afschrijvingsstaat	R	(Commerciële) Afschrijvingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Commerciële afschrijvingstermijn	R	(Commerciële) Afschrijvingstermijn raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Fiscale aanschaffingsstaat	R	Fiscale aanschaffingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Fiscale afschrijvingsstaat	R	Fiscale afschrijvingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Fiscale afschrijvingstermijn	R	Fiscale afschrijvingstermijn				3	3	6	2	3,0	1,2
Vaste activa	Investering	R	Investering van Actief weergeven				3	3	6	2	3,0	1,2
Vaste activa	Actief	U	Actief bewerken Financieel				5		5	1	5,0	1,0
Algemeen	Bijlage	C	Bijlage toevoegen aan inkoopfactuur				5		5	1	5,0	1,0
Dossier	Dossieritem	D	Dossieritem verwijderen	1	1	1	1	1	5	5	1,0	1,0
Vaste activa	Actief	D	Actief verwijderen				4		4	1	4,0	0,8
Vaste activa	Factuur	C	Factuur aan Actief toevoegen				4		4	1	4,0	0,8
Journaalpost	Journaalpost	UD	Journalisering van (geselecteerde) Activa terugdraaien				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	C	Subsidie aan Actief toevoegen				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	R	Subsidie van Actief weergeven				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	U	Subsidie van Actief bewerken				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	D	Subsidie van Actief verwijderen				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	C	Actief verkopen				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	R	Actief verkoop weergeven				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	U	Actief verkoop bewerken				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	D	Actief verkoop verwijderen				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	C	Verzekering toevoegen				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	R	Verzekering weergeven				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	U	Verzekering bewerken				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	D	Verzekering verwijderen				2	2	4	2	2,0	0,8
Vaste activa	Investering	C	Investering aan Actief toevoegen				3		3	1	3,0	0,6
Vaste activa	Investering	U	Investering van Actief bewerken				3		3	1	3,0	0,6
Vaste activa	Investering	D	Investering van Actief verwijderen				3		3	1	3,0	0,6
Vaste activa	Actief	U	Verzekering koppelen aan actief				2		2	1	2,0	0,4
Vaste activa	Factuur	U	Factuur van Actief bewerken				2		2	1	2,0	0,4
Vaste activa	Actief	R	Rapportage Controleoverzicht vaste activa boekhouding raadplegen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	C	Type subsidie toevoegen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	C	Locatie actief toevoegen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	R	Type subsidie weergeven				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	R	Locatie actief weergeven				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	U	Type subsidie bewerken				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	U	Locatie actief bewerken				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	D	Type subsidie verwijderen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	D	Locatie actief verwijderen				1	1	2	2	1,0	0,4
Vaste activa	Factuur	D	Factuur van Actief verwijderen				1		1	1	1,0	0,2

Figure 5.6: REPORT of case study *Fixed assets in AFAS InSite*

The results of the case study have been received well by the case company, as the product managers had not yet rationalized the mapping of functionality of the function group *Fixed assets*. Especially since the previous case study had not revealed new insight in future work, but was rather more focused on the validation of the template method. During the case study and mapping group session, it became clear how the template method is limited in its contribution to the product roadmap, as it does not take into account the required and available resources for the implementation of the mapped functionality on another software platform. Also, it became evident that there are strong relationships between sets of functionality, which may influence the priority of mappings, but the template method does not provide tools to cascade the priority of a mapping to related functionality.

The participants of the group session explained how the case study had helped them to think about the function group *Fixed assets* in AFAS InSite in a strict way, given the fact that every bit of functionality had to be rationalized for every scenario, and an average priority would roll out automatically, based on their assessment.

Chapter 6

Discussion

During the course of this research project, it has become evident that both the research itself and the designed artifacts have their limitations. We believe the explication of these limitations is not a sign of weakness, but rather a strength of the research, in the sense that it provides new opportunities for future research and continuation of the research subject.

The research project's goal is to design a template method for software developing organizations. Since the role of a software product manager does not necessarily imply having in-depth knowledge about the product's source code and technical architecture, techniques that concern technical competencies such as the analysis of source code or implemented architecture, are omitted. Thus, the software product is analyzed from a functional perspective. However, this might not be conclusive, and a functional approach may produce more overhead and consume more resources compared to those technical, potentially automated approaches that were left out in the research.

During the extraction of entities and functionality from the software product, functionality is not clustered, nor are relationships between functionality recorded. Should we have decided to do so, the process of mapping functionality on scenarios would have become too complex, due to a cascading assignment of priorities amongst functionality. This does not benefit the efficiency of the template method, as the mapping phase is not designed to consider such extensive dependencies.

The template method's deliverable, an instantiation of the concept REPORT, is not suitable to be considered a product roadmap. The method does not take into account the required and available resources for the implementation of each set of functionality during the mapping phase. The required resources is not exclusively dependent on the characteristics of the functionality itself, as the difficulty of implementing functionality can be different per software platform. However, it has been a conscious decision to exclude the consideration of resources, as it would increase the complexity of the mapping phase, making the method less efficient. The prioritized shortlist of requirements, delivered by the method, can in turn be used as input for roadmap intelligence.

The template method has been designed, instantiated and validated at a single case company, which produces an integrated ERP software product. It may be possible that validation at another case company, producing different software products, possibly even adhering to another development methodology, may result in different performance. Such validation is left open for future work.

In the case studies of this research project, an existing software product has been examined, which is AFAS Profit. In order to evaluate the evolution of the software product, the software platforms AFAS InSite and AFAS OutSite were included in the template method instantiations. However, those platforms are also already in their production phase, yet they do resemble a significant part of the functionality in the AFAS Profit Windows software suite. Since the software platforms in scope are already live and have already been developed, the case studies did not explore the template method's performance in cases of virtual, non-existent software platforms that are still to be developed. Therefore, evaluation of the template method is subject to future work in case studies where software platforms are considered for which an application has not yet been developed.

The case studies of the research have both focused at all functionality within one function group of the software product. The function groups to be analyzed have been pointed out by a software product manager of the case company, given the relevance in the research project and the personal interests of the product managers concerning the mapping of the function groups. This implies that the template method has not been instantiated for a complete software product, nor has it been instantiated for multiple function groups.

Also, given the limitation in the previous paragraph, it has not been explored whether the template method can decide what function groups to implement first on new software platforms first, thus maintaining a higher level of abstraction on the software's functionality.

Concerning the representation of the results of a template method instantiation, the current visualization focuses mainly around a matrix of functionality, scenarios and mappings. Even though the visualization is not to be designed as a product roadmap, the participants of the case company have also indicated that the research could have done more on visualization of the deliverables. However, it is not certain whether it would be possible to visualize mappings as (for instance) a network, since no relationships between functionality are captured by the method. We do acknowledge that the reporting aspect of the template method leaves room for future research.

Chapter 7

Conclusion

In this research, the Software Functional Evolution Method (SFEM) has been proposed, a structured approach towards the evolution of a software product by mapping functionality between software platforms. The template method is designed for software product managers, having a functional perspective on the analysis of existent software products and their functionality.

In Section 1.3, the main research question was established as follows: **What method assists in software product evolution through mapping of functionality between software platforms?** We answer this main research question by first answering the four sub questions, followed by an answer to the main research question.

1. **Which methods assist in the identification and characterization of software functionality from a functional perspective?**

   We have identified five techniques to extract functionality from an existent software product. First, by analysis of the software product's user manual, the user processes for which the software has been designed, become clear, revealing the underlying functionality. Second, formal design artifacts contain detailed information about the expected functioning of the product, and how this has been implemented in the product. Analysis of descriptive texts of the software's functionality can be assisted by Natural Language Processing. The graphical user interface of the software product reveals end-user functionality and features by interaction elements. Finally, architecture reconstruction extracts the implemented software architecture from the software product, which can be used to define how functionality links together different entities within the product.

   Different functionality classifications are described, which can help in an efficient identification of the majority of functionality, after the entities in scope have been identified. The *Create*, *Read*, *Update*, *Delete* (CRUD) classification can be extended with a *Scan* operator, of which the BREAD classification is similar. Finally, the Read-only / Maintain classification is a simplified classifier of software functionality.

2. **What characterizes the users and functional context and constraints of a software platform?**

   The research has explored the characterization of users by personas in four focus areas. First, the characteristics of a persona assist in making the persona live in the minds of the designers. Second, needs and goals explicate what the persona wishes to accomplish, which can be achieved by the use of the software product. Third, skills and competencies define what a persona is able to do, but also what constrains him/her in performing certain actions. Last, explicit recording of the persona's constraints reveals what may limit the mapping of functionality, based on the characteristics of the persona.

Software platforms are defined in the research by four focus areas. First, two typologies of the platform type are given, which are *desktop*, *web*, *mobile* or *wearable*, and *internal platform*, *supply chain platform*, *industry platform* or *two-sided market*. A SWOT analysis identifies how a software platform can contribute to strategic roadmapping of the product. The functional context and constraints reveal what may limit the mapping of functionality, as well as the technical context and constraints.

3. **Which methods assist in the prioritization of functionality when mapping functionality between software platforms?**

   Given the importance of the priority that is assigned to a mapping of functionality between software platforms, eight different prioritization techniques are described and compared by means of their *scale*, *granularity* and *sophistication*. The techniques include the *binary priority list*, *priority groups*, *MoSCoW*, *requirements triage*, *cumulative voting*, *ranking*, *top ten* and the *Wieger's prioritization model*.

   The concept "mapping" is introduced as being the decision to implement a set of functionality on another software platform. The likely occurrence of a persona on a software platform is defined as a scenario. The instantiation of a mapping connects functionality to a given scenario.

4. **How can the results of a method instantiation be reported in order to assist in the evolution of a software product?**

   It has become clear that the results of an instantiation cannot serve as a product roadmap on its own. Therefore, the report of an instantiation serves as input for product roadmapping, basically being presented as an ordered list of product requirements for new applications on new software platforms. We have introduced the mapping matrix as the most obvious way to report an instantiation, although the choice of medium depends on the stakeholder audience. A categorization of requirements can be applied based on the related component, product, industry or release.

Having answered the sub questions of the research project, we can answer the main research questions, which was stated:

What method assists in software product evolution through mapping of functionality between software platforms?

This research proposes the Software Functionality Evolution Method, a template method which assists a software developing organization in the evolution of a software product by means of mapping functionality between software platforms. The method consists of five main activities, which are basically (1) the defining of the project to be performed, (2) the extraction of entities and functionality from the software product, (3) the identification of personas and software platforms in scenarios, (4) the mapping and prioritizing of functionality on scenarios, and (5) the reporting of the project's outcome to selected stakeholders. A variety of theoretical foundations has been given, which support the instantiation of the activities and the production of the method's concepts.

The Software Functionality Evolution Method has been designed by means of an incremental method engineering approach. The template method has been instantiated in two case studies, which has been recorded in a backlog. By analyzing the template method and the instantiation's backlog, method increments have been designed to improve the template method. In order to reflect on the incremental approach, a categorization of *constructing*, *labeling* and *associating* has been proposed. The categorization allows for reasoning and discussion of the method evolution.

A correct application and instantiation of the template method can help organizations to embrace opportunities resulting from emerging technologies and modern software platforms. A more rapid and rational response to designing requirements for new applications leads to a competitive advantage, compared to other market players who struggle with efficiency in their Software Product Management practices.

We believe that this research project contributes to the many attempts to bridge the gap between scientific literature and industrial practices in the field of Software Product Management. A common understanding of the theoretical foundations and actual practices can assist in a better education of software product managers, which are of growing importance in the modern industry of software vendors.

7.1 Future research

The two case studies in which the template method is instantiated are performed at one single software developing organization in the Netherlands, which produces an integrated ERP system for the Dutch marketplace. Future work will further validate and improve the template method by means of case studies at other software vendors, with different software products and different project requirements. Thus, a more quantitative approach towards the instantiation and validation of the template is subject to future research.

Also, concerning the validation and evaluation of the template method, the status of the software platforms in the case studies has been discussed in Chapter 6. Future work will explore the template method performance in cases where functionality is to be mapped between software platforms for which an application has not yet been developed. This increases the level of abstraction about the software platform, and may or may not have an impact on the method's performance.

Another example is the mapping of functionality in a case study where there is not even a software product to extract functionality from. Future research will explore if and how the method performs in case of a software product which still has to be designed and developed, and where the central question is which functionality should be designed and implemented for which software platform.

The research project is designed for software product managers, and thus technical approaches for the analysis of software products and their architecture are omitted. Future work will explore the (semi-)automated analysis of software products, to make the process of extracting entities and functionality more efficient and conclusive.

In the mapping phase of the template method, opportunities exist for automated application and cascading of priorities, based on relationships between entities and functionality. However, this is not explored in this research project, as it is still unclear what effect cascading has on the outcome of an instantiation. Such dependencies would require a thoroughly tested algorithm which automatically cascades a mapping's priority based on properties of the relationship between entities or functionality.

Future research can also expand on the visualizations for reporting the results of a template method instantiation. Currently, the mapping matrix serves the goal of providing input to product roadmapping, yet it is uncertain whether the resulting mappings can be clustered to create networks of functionality, or any other relevant visualization technique.

Bibliography

Ahl, V. (2005). *An Experimental Comparison of Five Prioritization Methods*. Master's thesis, Blekinge Institute of Technology.

Aoyama, M. (2005). Persona-and-Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 85–94.

Baldwin, C. Y. and Woodard, C. J. (2009). *The Architecture of Platforms: A Unified View*. Edward Elgar London.

Bebensee, T., Van de Weerd, I., and Brinkkemper, S. (2010). Binary Priority List for Prioritizing Software Requirements. In *Requirements Engineering: Foundation for Software Quality*, pages 67–78. Springer Berlin Heidelberg.

Bekkers, W., Van de Weerd, I., Spruit, M., and Brinkkemper, S. (2010). A Framework for Process Improvement in Software Product Management. In *Systems, Software and Services Process Improvement*, volume 99, pages 1–12. Springer Berlin Heidelberg.

Berander, P. (2004). Using Students as Subjects in Requirements Prioritization. In *International Symposium on Empirical Software Engineering*, pages 167–176.

Berander, P. and Andrews, A. (2005). Requirements Prioritization. In *Engineering and Managing Software Requirements*, pages 69–94. Springer Berlin Heidelberg.

Bisbal, J., Lawless, D., Wu, B., and Grimson, J. (1999). Legacy Information Systems: Issues and Directions. *Software, IEEE*, 16(5):103–111.

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Number September. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA.

Bosch, J. (2009). From Software Product Lines to Software Ecosystems. In *Proceedings of the 13th International Software Product Line Conference*, number Splc, pages 111–119. Carnegie Mellon University.

Bresnahan, T. F. and Greenstein, S. (1999). Technological Competition and the Structure of the Computer Industry. *The Journal of Industrial Economics*, 47(1):1–40.

Brinkkemper, S. (1996). Method Engineering: Engineering of Information Systems Development Methods and Tools. *Information and software technology*, 38(4):275–280.

Brodie, M. L. and Stonebraker, M. (1995). *Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Carlshamre, P. (2002). Release Planning in Market-Driven Software Product Development: Provoking an Understanding. *Requirements Engineering*, 7(3):139–151.

Chan, F., Chan, M., and Tang, N. (2000). Evaluation Methodologies for Technology Selection. *Journal of Materials Processing Technology*, 107(1-3):330–337.

Chowdhury, G. G. (2003). Natural Language Processing. *Annual Review of Information Science and Technology*, 37(1):51–89.

Cole, M. and Avison, D. (2007). The Potential of Hermeneutics in Information Systems Research. *European Journal of Information Systems*, 16(6):820–833.

Colomo-Palacios, R., Fernandes, E., Soto-Acosta, P., and Sabbagh, M. (2011). Software Product Evolution for Intellectual Capital Management: The Case of Meta4 PeopleNet. *International Journal of Information Management*, 31(4):395–399.

Condon, D. (2002). *Software Product Management: Managing Software Development from Idea to Product to Marketing to Sales*. Aspatore Books.

Cooper, A. (1999). *The Inmates are Running the Asylum: Why Hightech Products Drive us Crazy and How to Restore the Sanity*. SAMS, Macmillan Computer Publishing, Indianapolis, IN.

Cooper, A., Reimann, R., and Cronin, D. (2012). *About Face 3: The Essentials of Interaction Design*. John Wiley & Sons.

Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., and Sears, R. (2010). Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154, New York, New York, USA. ACM, ACM Press.

Danilovic, M. and Sandkull, B. (2005). The Use of Dependence Structure Matrix and Domain Mapping Matrix in Managing Uncertainty in Multiple Project Situations. *International Journal of Project Management*, 23(3):193–203.

Davis, A. M. (2003). The Art of Requirements Triage. *Computer*, 36(3):42–49.

DSDM Consortium (2008). *DSDM Atern Handbook*. DSDM Consortium.

Dver, A. S. (2003). *Software Product Management Essentials*. Anclote Press, Tampa, Florida.

Ebert, C. (2007). The Impacts of Software Product Management. *Journal of Systems and Software*, 80(6):850–861.

Gawer, A. (2009). *Platform Dynamics and Strategies: From Products to Services*. Cheltenham: Edward Elgar Publishing Limited.

Gawer, A. and Cusumano, M. A. (2003). Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation. *Innovation: Management, Policy & Practice*, 5(1):91–94.

Gimnich, R. and Winter, A. (2005). Workflows der Software-Migration. *Softwaretechnik-Trends*, 25(2):22—-24.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220.

Harmsen, F., Brinkkemper, S., and Oei, H. (1994). *Situational Method Engineering for Information System Project Approaches*. Number September. University of Twente, Department of Computer Science.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. *MIS quarterly*, 28(1):75–105.

ISPMA (2014). *Software Product Management Body of Knowledge*. http://ispma.org/spmbok/.

Jung, C. G. and Storr, A. E. (1983). *The Essential Jung*. Princeton University Press.

Junior, P. T. A. and Filgueiras, L. V. L. (2005). User Modeling with Personas. In *Proceedings of the 2005 Latin American conference on Human-computer interaction*, pages 277–282, New York, New York, USA. ACM Press.

Karlsson, J. and Ryan, K. (1997). A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, 14(5):67–74.

Karlsson, J., Wohlin, C., and Regnell, B. (1998). An Evaluation of Methods for Prioritizing Software Requirements. *Information and Software Technology*, 39(14):939–947.

Kazman, R. and Carrière, S. J. (1999). Playing Detective: Reconstructing Software Architecture from Available Evidence. *Automated Software Engineering*, 6(2):107–138.

Kazman, R., O'Brien, L., and Verhoef, C. (2003). Architecture Reconstruction Guidelines, Third Edition. Technical Report November, Software Engineering Institute, Carnegie Mellon University.

Knuth, D. E. (2005). *The Art of Computer Programming*. Pearson Education.

Lee, J. (1997). Design Rationale Systems: Understanding the Issues. *IEEE Expert: Intelligent Systems and Their Applications*, 12(3):78–85.

Leffingwell, D. and Widrig, D. (2003). *Managing Software Requirements: A Use Case Approach*. Addison-Wesley Professional.

Lehtola, L., Kauppinen, M., and Kujala, S. (2005). Linking the Business View to Requirements Engineering: Long-Term Product Planning by Roadmapping. In *13th IEEE International Conference on Requirements Engineering (RE'05)*, pages 439–443. Ieee.

Mann, S. (1997). Wearable Computing: A First Step Toward Personal Imaging. *Computer*, 30(2):25–32.

Martin, J. (1983). *Managing the Data Base Environment*. Prentice Hall PTR.

Meyer, M. H. and Lehnerd, A. P. (1997). *The Power of Product Platforms: Building Value and Cost Leadership*. Free Press, New York.

Moher, D., Liberati, A., Tetzlaff, J., and Altman, D. G. (2009). Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement. *Annals of Internal Medicine*, 151(4):264–269.

Moon, M., Yeom, K., and Chae, H. S. (2005). An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line. *IEEE Transactions on Software Engineering*, 31(7):551–569.

Muffatto, M. and Roveda, M. (2002). Product Architecture and Platforms: A Conceptual Framework. *International Journal of Technology Management*, 24(1):1–16.

Müller, H. A. and Klashinsky, K. (1988). Rigi-A system for Programming-In-The-Large. In *Proceedings of the 10th International Conference on Software Engineering*, pages 80–86. IEEE Computer Society Press.

Object Management Group (2004). UML 2.0 Superstructure Specification. Technical report, Technical Report ptc/04-10-02.

O'Brien, L., Stoermer, C., and Verhoef, C. (2002). Software Architecture Reconstruction: Practice Needs and Current Approaches. Technical Report August, Software Engineering Institute, Carnegie Mellon University.

Phaal, R., Farrukh, C. J. P., and Probert, D. R. (2004). Technology Roadmapping - A Planning Framework for Evolution and Revolution. *Technological forecasting and social change*, 71(1):5–26.

Pohl, K., Böckle, G., and Van der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, Heidelberg.

Pruitt, J. and Grudin, J. (2003). Personas: Practice and Theory. In *Proceedings of the 2003 conference on Designing for user experiences*, pages 1–15.

Racheva, Z., Daneva, M., and Buglione, L. (2008). Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products. In *Second International Workshop on Software Product Management*, pages 49–58, Barcelona, Catalunya.

Rajlich, V. T. and Bennett, K. H. (2000). A Staged Model for the Software Life Cycle. *Computer*, 33(7):66–71.

Regnell, B. and Brinkkemper, S. (2005). Market-Driven Requirements Engineering for Software Products. In *Engineering and Managing Software Requirements*, pages 287–308. Springer Berlin Heidelberg.

Regnell, B., Höst, M., Och Dag, J. N., Beremark, P., and Hjelm, T. (2001). An Industrial Case Study on Distributed Prioritisation in Market- Driven Requirements Engineering for Packaged Software. *Requirements Engineering*, 6(1):51–62.

Robertson, D. and Ulrich, K. (1998). Planning for Product Platforms. *Sloan management review*, 39(4).

Robertson, P. (1997). Integrating Legacy Systems with Modern Corporate Applications. *Communications of the ACM*, 40(5):39–46.

Rochet, J.-C. and Tirole, J. (2003). Platform Competition in Two-Sided Markets. *Journal of the European Economic Association*, 4(1):990–1029.

Rogers, Y., Sharp, H., and Preece, J. (2011). *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons.

Royce, W. W. (1970). Managing the Development of Large Software Systems. In *proceedings of IEEE WESCON*, number August, pages 1–9, Los Angeles.

Ruhe, G. and Saliu, M. O. (2005). The Art and Science of Software Release Planning. *IEEE Software*, 22(6):47–53.

Runeson, P. and Höst, M. (2009). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14(2):131–164.

Rysman, M. (2009). The Economies of Two-Sided Markets. *The Journal of Economic Perspectives*, pages 125–143.

Sommerville, I. (2007). *Software Engineering*. Addison-Wesley, 8th edition.

Sommerville, I. and Sawyer, P. (1997). *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Inc.

Stevens, S. S. (1946). On the Theory of Scales of Measurement. *Science (New York, N.Y.)*, 103(2684):677–80.

Stolze, M., Riand, P., Wallace, M., and Heath, T. (2007). Agile Development of Workflow Applications with Interpreted Task Models. In *6th international conference on Task Models and Diagrams for User Interface Design*, pages 2–14. Springer.

The Standish Group (1995). *The CHAOS Report*. Project Smart.

Van Angeren, J. (2013). *Exploring Platform Ecosystems: A Comparison of Complementor Networks and their Characteristics*. PhD thesis, Utrecht University.

Van de Weerd, I. and Brinkkemper, S. (2008). Meta-Modeling for Situational Analysis and Design Methods. In *Handbook of research on modern systems analysis and design technologies and applications*, volume 35, pages 35–54. Information Science Reference.

Van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., and Bijlsma, L. (2006). On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. In *14th IEEE International Requirements Engineering Conference*, pages 319–322. Ieee.

Van de Weerd, I., Brinkkemper, S., and Versendaal, J. (2007). Concepts for Incremental Method Evolution: Empirical Exploration and Validation in Requirements Management. In Krogstie, J., Opdahl, A., and Sindre, G., editors, *Advanced Information Systems Engineering SE - 33*, volume 4495 of *Lecture Notes in Computer Science*, pages 469–484. Springer Berlin Heidelberg.

Van den Akker, M., Brinkkemper, S., Diepen, G., and Versendaal, J. (2008). Software Product Release Planning Through Optimization and What-If Analysis. *Information and Software Technology*, 50(1-2):101–111.

Van der Schuur, H. (2011). *Process Improvement Through Software Operation Knowledge — If the SOK Fits, Wear It!* {PhD} dissertation, Utrecht University.

Van der Schuur, H., Jansen, S., and Brinkkemper, S. (2011). If the SOK Fits, Wear It: Pragmatic Process Improvement Through Software Operation Knowledge. In *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*, pages 306–321. Springer.

Wiegers, K. (1999a). First Things First: Prioritizing Requirements. *Software Development*, 7(9):48–53.

Wiegers, K. (1999b). *Software Requirements*. Microsoft Press (Redmond, WA).

Xu, L. and Brinkkemper, S. (2005). Concepts of Product Software: Paving the Road for Urgently Needed Research. In *The first International Workshop on Philosophical Foundations of Information Systems Engineering*, pages 523–528.

Yee, J. T. and Oh, S.-C. (2013). Focusing on RFID, Interoperability, and Sustainability for Manufacturing, Logistics, and Supply Chain Management. In *Technology Integration to Business*, pages 67–95. Springer.

Yin, R. K. (2009). *Case Study Research: Design and Methods*. Sage.

List of figures

List of tables

Appendices

Appendix A

Paper IWSPM14

This paper has been submitted and accepted to the 8th International Workshop on Software Product Management (IWSPM 2014). The workshop is co-located with the 22nd IEEE International Conference on Requirements Engineering in Karlskrona, Sweden, from August 25 to 29, 2014.

Software product management (SPM) is an important discipline that unites both technical and business perspectives to creating software products. The success of a product depends on skilled and competent product management. In essence, a product manager makes strategic and tactical decisions on what functionality and quality a product should offer, to which customers, and at what time. IWSPM'14 aims at contributing to the body of knowledge for software product management in order to support the evolution of software product management as a scientific discipline and as a practical approach to management of software products. The workshop should also foster collaboration between academia and industry. The workshop main goals are:

- Contribute to the SPM Body of Knowledge (SPMBoK);
- Identify challenges and future avenues for research relevant for SPM practice;
- Strengthen SPM as a research field within the greater field of software engineering and business management;
- Provide software product managers and researchers a dedicated form for exchanging ideas and best practices fostering industry-academia collaboration;
- Offer opportunities for industry participants to contribute with articles and participation.

Important dates

- Paper Submission: Friday, June 6, 2014
- Paper Notification: Monday, June 23, 2014
- Camera Ready Due: Monday, July 7, 2014
- Workshop: Tuesday, August 26, 2014

Bridging the Gap Between Software Platforms: A Template Method for Software Evolution

Gerard Nijboer
Department of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
g.nijboer@students.uu.nl

Henk van der Schuur
AFAS Software
Leusden, The Netherlands
h.vdschuur@afas.nl

Jan Martijn E.M. van der Werf
Department of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
j.m.e.m.vanderwerf@uu.nl

Sjaak Brinkkemper
Department of Information and Computing Sciences
Utrecht University
Utrecht, The Netherlands
s.brinkkemper@uu.nl

***Abstract*—To prevent issues arising from legacy software platforms, adapting to changing customer needs by software evolution is a growing concern of software organizations. However, current practices are pragmatic and subjective, which restricts benchmarking and reduces efficiency. In order to improve evolutionary practices, this paper proposes the Software Functionality Evolution Method (SFEM). The SFEM provides a software vendor with input for product roadmapping, by mapping functionality between software platforms. Mappings are based on characteristics and constraints of functionality, personas and software platforms. An incremental method engineering approach is put to practice, in which the template method is instantiated and improved over multiple case studies. Case studies show that the method contributes to efficient reasoning and strategic decision making in software evolution for software product managers.**

I. Introduction

The current pace of technological developments offers opportunities to software developing organizations concerning software platforms and applications. In order to avoid issues caused by legacy software platforms, concerning costs, maintenance, accessibility and extensibility [1], emerging technologies can help organizations to innovate, improve efficiencies, and realize new business opportunities [2]. Software evolution concerns the adaption of capabilities and functionality of a system, in order to meet user needs [3].

Currently, no structured approach exists which assists in the evolution of a software product by mapping functionality on new software platforms. Thus, a gap exists in the evolutionary process, as a mapping of functionality between platforms needs to be created, yet it is uncertain what functionality should be included. A structured approach enables comparison of performance and results, and increases efficiency in method instantiations.

Considering the problem statement above, the main research question of this paper is: *What method could assist in software product evolution through mapping functionality between software platforms?*

This paper proposes a method, the Software Functionality Evolution Method (SFEM), which assists in the evolution of a software product. It is designed for software product managers, as this organizational role is responsible for strategic decision making [4], including a software product's lifecycle [5]. With this audience in mind, the evolutionary process is considered with a focus on functional, rather than technical properties and constraints. The constraints, raised by characteristics of personas, platforms and functionality, determine the mapping and priority of functionality.

The SPM Competence Model [5] proposes 15 focus areas in the field of Software Product Management (SPM) practices. To position this research and the designed method in the field of SPM, Figure 1 visualizes the relationships with the different focus areas and competencies. Three categories are applied to related focus areas: (1) *triggers* which instantiate the method, (2) *execution* for the mutual support of activities, and (3) *output* for those focus areas that have an interest in the results of an instantiation.

An instantiation of the method can be triggered by activities within the focus areas *Market analysis* and *Product lifecycle management*. An opportunity can be identified, such as a new software platform, which generates a competitive advantage for the software company if implemented correctly.

The SFEM assists in the execution of activities within the focus areas *Requirements gathering*, *Requirements organizing* and *Requirements prioritization*. By mapping existing functionality between platforms, requirements are gathered and prioritized based on their added value in the market and product portfolio.

The results of an instantiation of the method provides the organization with information which can be used in activities in the focus areas *Release definition*, *Roadmap intelligence* and *Product roadmapping*. On the short term, mappings of functionality help to identify which requirements add significant value to a new release. On the long term, mappings assist in the creation of themes for the product roadmap.

This introduction is followed by an explanation of the research approach in Section II. The Software Functionality Evolution Method is presented in Section III. In Section IV, a categorization for method increments is presented, which enables reflection on the process of incremental method engineering. The results of the method instantiations in case studies are presented in Section V. Section VI contextualizes the research with related literature, followed by a discussion

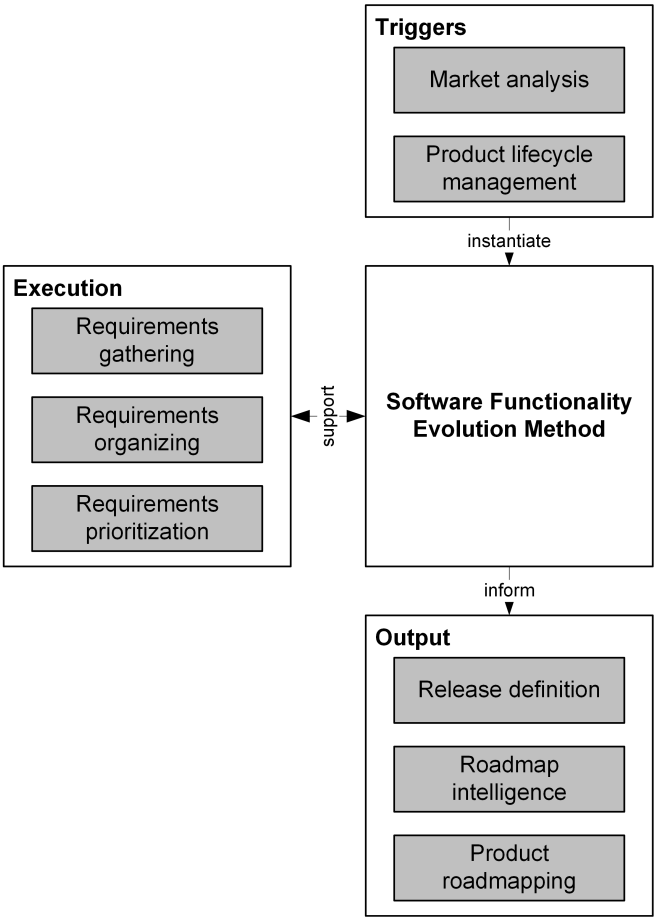


Fig. 1. Positioning of the research to the SPM Competence Model

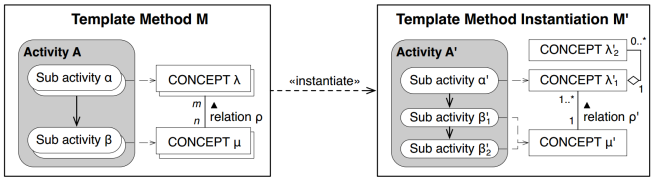


Fig. 2. Template method instantiation [10]

in Section VII and the conclusion in Section VIII.

II. Research approach

The research is based on a combination of a literature review and interviews with domain experts at a Dutch Enterprise Resource Planning (ERP) software vendor. The approach followed in the literature review is inspired by the PRISMA 2009 checklist [6].

By means of method engineering [7]–[9], the initial research results in a conceptual, initial version of the method, which is called the Software Functionality Evolution Method (SFEM). Different from a situational method [7], [8], this template method prescribes what activities and concepts are to be implemented, rather than what an instantiation of the method would look like [10].

In two case studies at the ERP software vendor, the template method is instantiated, of which a backlog is recorded for analysis purposes. This backlog contains rationales on the instantiation of activities and concepts, and decisions made accordingly. A structured approach towards the case studies is followed [11], [12].

The performance of the template method instantiation is analyzed, in order to identify opportunities to improve the method. The cycle of method engineering, template method instantiation and method improvement is repeated until a stable version of the SFEM is engineered. The method will be contributed to the Software Product Management Body of Knowledge (SPMBOK) [13], classifying the research as design science [14].

The results of the initial research lay a basis for the design of the first version of the template method. The template method and a case study instantiation are analyzed, which results in a set of method increments to improve the template method. The process of instantiation and analysis is repeated in a second case study, which results in the final template method.

III. Software Functionality Evolution Method

The Software Functionality Evolution Method (SFEM) is a template method which assists a software developing organization in the evolution of a software product by means of mapping functionality between software platforms. A template method is different from a situational method as it serves as a template for an instantiation, rather than describing the instantiation of the situational method itself. In Figure 2, the concept of template method instantiation is visualized [10]. The figure indicates how the open activities and concepts of a template method may result in extra elements after instantiation.

The method is visualized as a Process-Deliverable Diagram (PDD), which is a technique used for modeling activities and artifacts of a certain process [9]. On the left side of the diagram are the activities of the method's process, of which the notation is based on the UML activity diagram [15]. On the right side of the diagram, deliverables are visualized as concepts to indicate what artifacts are produced by a template method instantiation, of which the notation is based on the UML class diagram [15].

The template method's PDD is shown in Figure 3. The corresponding concepts are explained in Section III-A, followed by theoretical foundations in Section III-B.

A. Concepts

The concepts of the SFEM are the artifacts of a template method instantiation, produced by the execution of activities in the method. We introduce a definition of each concept in the method, followed by theoretical foundations in Section III-B.

PROJECT PLAN — The PROJECT PLAN is a document that describes the technical and management approach to be followed for a project. The plan typically describes the work

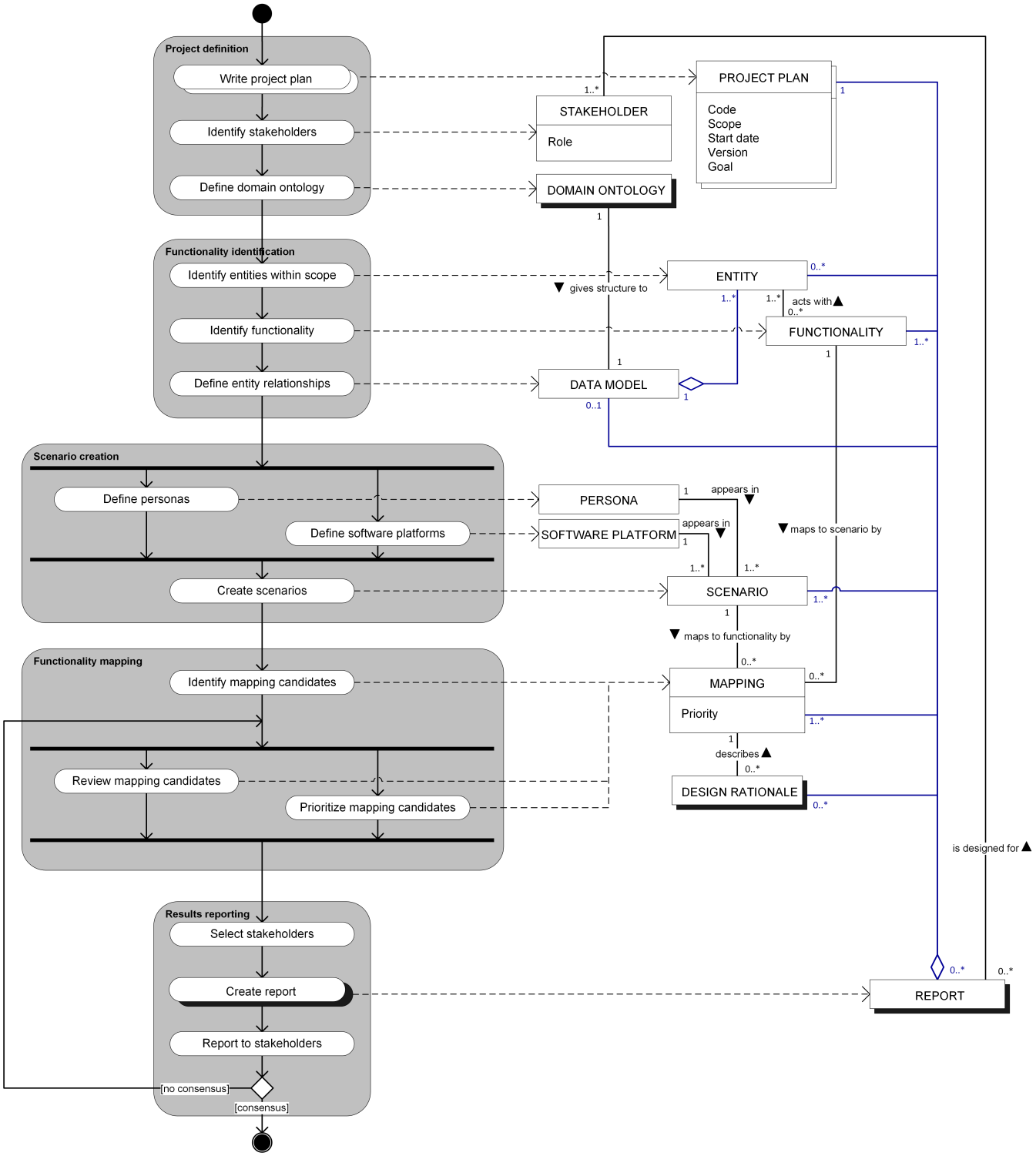


Fig. 3. Software Functionality Evolution Method as a Process-Deliverable Diagram

to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way that the project will be organized [16].

STAKEHOLDER — A STAKEHOLDER is an individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations [17]. A STAKEHOLDER's *Role* in a template method instantiation can be organized as described in Section III-B1.

DOMAIN ONTOLOGY — An ontology describes the ENTITIES within the domain in discourse, and how these ENTITIES are interrelated [18]. The DOMAIN ONTOLOGY represents the domain in which the software product is designed to operate, the domain of discourse.

ENTITY — In computer programming, an ENTITY is any item that can be named or denoted in a program. For example, a data item, program statement, or subprogram. [16].

FUNCTIONALITY — FUNCTIONALITY concerns the capabilities of the various computational, user interface, input, output, data management, and other features provided by a product [19].

DATA MODEL — A DATA MODEL identifies the ENTITIES, domains (attributes), and relationships (associations) with other data and provides the conceptual view of the data and the relationships among data [20].

PERSONA — PERSONAS are defined as representations of the actual users of a system, defined by the goals they aim to accomplish. They are hypothetical archetypes of actual users [21].

SOFTWARE PLATFORM — A platform is the combination of an operating system and hardware that makes up the operating environment in which a program runs [22]. Thus, a SOFTWARE PLATFORM defines the environment in which a software product is designed to operate.

SCENARIO — The combination of possible and relevant appearances of PERSONAS on SOFTWARE PLATFORMS creates an instantiation of the concept SCENARIO. A SCENARIO is used to map FUNCTIONALITY by MAPPINGS and their *Priorities*.

MAPPING — A MAPPING is an assigned correspondence between two things that is represented as a set of ordered pairs [20]. The concepts is instantiated by the combination of a SCENARIO with FUNCTIONALITY. The *Priority* of a MAPPING is determined based on the importance of the FUNCTIONALITY for the given SCENARIO. If a MAPPING has no *Priority* assigned, it is considered to be a candidate.

DESIGN RATIONALE — A DESIGN RATIONALE is defined as information capturing the reasoning of the designer that led to the system as designed, including design options, trade-offs considered, decisions made, and the justifications of those decisions [23]. It presents the arguments behind a MAPPING and its *Priority*.

REPORT — A REPORT is an information item that describes the results of activities such as investigations, observations, assessments, or tests [24]. The results of an instantiation are communicated to selected STAKEHOLDERS by a REPORT, which is designed to suit the STAKEHOLDER's interests.

		Degree of participation	
		Active	Passive
Degree of interaction	Direct	**Participant** - Project manager - Product manager - Domain expert	**Observer** - Project manager - Product manager - Board member
	Indirect	**Informer** - Domain expert - Customer - Partner	**Outsider** - Developer - Designer

Fig. 4. Method Stakeholder Classification Matrix

B. Theoretical foundations

To support the instantiation of activities and concepts of the SFEM, this research has explored various theoretical foundations. These foundations assist an analyst in the instantiation of the template method, by means of extra background information and supporting techniques for the implementation of processes.

1) Method Stakeholder Classification Matrix: In method engineering, it is possible that a process is explicitly carried out by a specific individual or organizational role. In that case, the role is indicated in the activity depicted in the method [9]. On the other hand, stakeholders are involved in a method instantiation to provide or consume information.

To help identify these stakeholders and apply a classification to their role in the method's instantiation, we introduce the Method Stakeholder Classification Matrix (MSCM), presented in Figure 4. The role of the stakeholder is dependent on the *degree of participation* in the instantiation of the template method, and the *degree of interaction* with the deliverables of the instantiation.

The MSCM is applicable in the activity *Identify stakeholders*, and makes the process of describing stakeholders more efficient. The selected value from the matrix can be used as the *Role* of the concept STAKEHOLDER.

2) Software functionality identification: To extract entities and their functionality from existent software products and underlying architectures, many techniques have already been discussed in scientific literature. Given the audience of this template method, as discussed in Section I, we have limited the exploration of such techniques by excluding technical approaches such as code-analysis. The techniques are applicable to the activities within the main activity *Functionality identification*, and the concepts resulting from these activities.

The analysis of a user manual allows for an analyst to identify functionality as it was designed and documented for the user. Natural Language Processing [25] can support analysis of such texts by tokenization, and generating tag clouds.

Architecture reconstruction, the process where the "as-built" architecture of an implemented system is obtained from an existing legacy system [26], helps in the identification of relationships among entities, and how this is translated into functionality. Different tools for reconstruction exist, such as ARMIN [26] and the Dali Architecture Reconstruction workbench [27].

A categorization of architecture reconstruction approaches [28] distinguishes *manual architecture reconstruction*, *manual reconstruction with tool support*, *query languages for writing patterns to build aggregations automatically*, and other techniques, such as *clustering*, *data mining* and *architecture description languages*.

To support the identification of functionality, the application of a functionality classifier to entities assists in covering a large portion of the functionality. Examples of such classifiers include CRUD [29], BREAD [30] and read-only/maintain.

3) Scenarios of personas and software platforms: In the template method, the concept SCENARIO plays a central role in the mapping of functionality between software platforms. As described in Section III-A, a scenario is the appearance of a persona on a given software platform. A persona may appear on one or more software platforms, and a software platform may host one or more appearances of personas. It is the combination of personas on platforms that is used to create the mapping of functionality, indicating the priority of the functionality for a given scenario.

As it is not desirable to let an actual user directly influence the designing process [21], the use of pretend users as personas is a good way to represent the actual users during systems design. Different sources [21], [31]–[33] have contributed to the following focus areas in the description of personas:

Characteristics — Make the persona live in the minds of designers by giving him/her characteristics like a name, photo, demographic data and attitudes.
Needs and goals — Explicate what a persona wishes to achieve, which can be achieved by the use of a software product. Goals can be classified as *life goals*, *experience goals* and *purpose goals* [34].
Skills and competencies — Driven by experience and knowledge, skills and competencies define a persona's abilities, and how they are limited in their actions.
Constraints — The separate definition of constraints, which may reside from other focus areas, puts extra emphasis on the inability of personas. These constraints are of particular interest when mapping functionality in the *Functionality mapping* activity phase of the template method.

Platforms, defined as "a foundation technology or set of components used beyond a single firm that brings together multiple parties for a common purpose or recurring problem" [35], represent opportunities for new software applications which a software vendor might adopt in the evolution of a software product. Four focus areas have been defined to help in the description of software platforms:

Platform type — A classification of the platform, either being *desktop*, *web*, *mobile* or *wearable* [36], [37] or *internal platform*, *supply chain platform*, *industry platform* or *two-sided market* [38].
SWOT analysis — The analysis of *strengths*, *weaknesses*, *opportunities* and *threats* of a software platform explicates the potential of the platform in the software product evolution.
Functional context and constraints — Given the functional context of a platform, either virtual or physical, it may enable or restrict the mapping of certain functionality.
Technical context and constraints — Technical opportunities or constraints may allow or disallow for the mapping of functionality to a software platform.

4) Mapping and prioritization: The mapping of functionality on scenarios is the main goal of the template method. The activities *Review mapping candidates* and *Prioritize mapping candidates* in the PDD of the SFEM are conducted in a group session with relevant stakeholders, in which the mapping and priority of functionality is discussed. A variety of requirements prioritization techniques exist [39], [40], of which the selection of the correct technique is dependent on the complexity of the project at hand.

Techniques which are relevant in the prioritization of requirements include the *Binary Priority List* [41], *cumulative voting* [42], *ranking* [39], the *Wiegers' prioritization model* [43], and *priority groups* [42], [44] such as *MoSCoW* [45] and *requirements triage* [46].

Creating a mapping is dependent on the characteristics and constraints of the persona and software platform of a scenario, and the characteristics and constraints of the regarded functionality. The priority assigned determines the importance of the mapping, compared to other mappings of functionality.

A mapping may have one or more instantiations of the concept DESIGN RATIONALE assigned, which captures the decision making process during the mapping activities. This allows for reasoning about the decisions in later stages of the product evolution.

IV. Reflecting on method increments

A multitude of methods has been developed since the emergence of the method engineering research field. All too often, however, the processes of method creation, as well as decisions made within remain underexposed, limiting understanding and repeatability of the respective method engineering research [10].

While elementary method increment types have been distinguished in earlier research [47], these types do not take into account reasoning and motivation for usage, limiting for reflection on method creation. We have attempted to address this issue by categorizing method increments. The following method increment categories[1] have been identified based on creation of the SFEM:

[1]Obviously, this set of method increment categories is not complete and is to certain extent specific to our method.

Constructing (C) — Adding, changing or removing (properties of) activities, concepts or properties in the diagram, including activity and concept types.
Labeling (L) — Adding, changing or removing a label of (properties of) activities, concepts, properties, associations or roles.
Associating (A) — Adding, changing or removing (properties of) associations between existing activities or concepts.

By explicating and motivating each increment in a method increment log, the method construction process as a whole is explicated. Below, an excerpt of the SFEM increment log with motivations per increment is shown.

C We need to capture documentation to reside with the VISUALIZATION. Instead of adding an explicit concept, a VISUALIZATION will include the concept DESIGN RATIONALE. This implies that a VISUALIZATION does not necessarily need to be a figure, it can just as well be textual.

C In the *Peer review* activity, a VISUALIZATION of the method's output must be included in order to review the performance. Therefore, we merged the *Peer review* activity with the main activity *Visualization*. This implies that after the *Visualization* activity, we must also be able to return to the *Mapping* main activity.

A We can't set the project's goal without knowing what the scope actually is we're working in. Therefore, the activity *Set project scope* will be implied by the activity *Define migration project*.

A The concepts SOFTWARE PLATFORM and PERSONA both appear in at least one SCENARIO, otherwise it would not make sense to define either of the concepts at all.

L The main activity *Platforms and personas* should be renamed to *Scenarios*, as the aim of the main activity is to develop SCENARIOS.

L The term for the concept OBJECT is ambiguous. The term ENTITY is more suitable, considering the method domain's jargon.

L The concept OBJECT TREE should be renamed to DATA MODEL, considering the method domain's jargon.

Figure 5 visualizes how different versions of template method instantiations contribute to the creation of the template method. Since a new version of a template method is constructed based on input from the previous version of the template method as well as its instantiation, template methods particularly benefit from method increment reflection. When a new version of a template method is to be developed, potential method increments (as well as underlying reasoning and motivations) are considered, compared and weighed from both an abstract/conceptual (template) perspective and a concrete/practical (instantiation) perspective. During the construction of the SFEM (Figure 3), we learned that this approach results in thorough yet rapid method development.

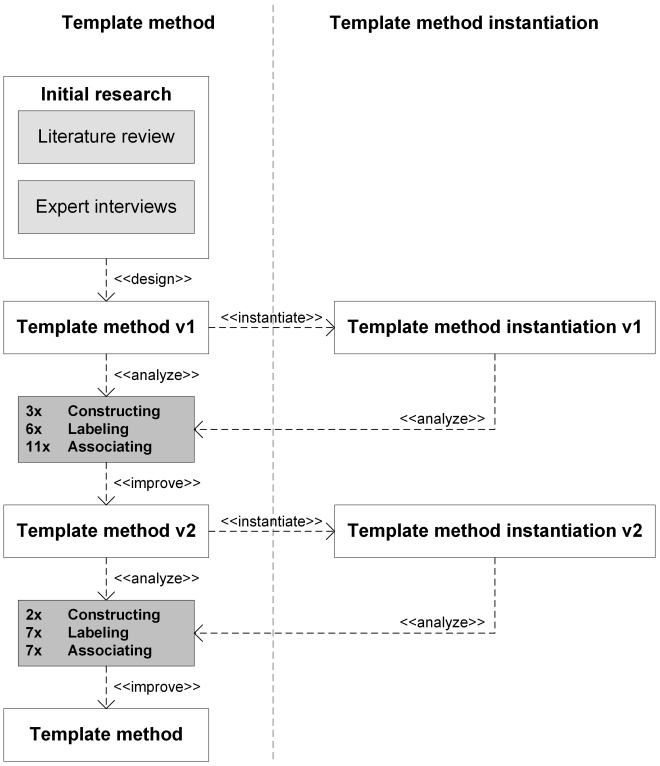


Fig. 5. Template method increments

V. CASE STUDY RESULTS

The Software Functionality Evolution Method (SFEM) has been instantiated in two case studies at an ERP software vendor, having its main office in the Netherlands. The software suite in scope exists of a Windows client application and two web applications, an intranet and a portal application. The Windows client is considered being the originating platform, having the base set of functionality.

In an ongoing attempt to evolve the software suite to meet current customer demands, functionality is being extracted from the Windows client and implemented in the web applications. The SFEM has been instantiated to assist the mapping of functionality of two function groups on the intranet platform.

In Figure 6, we present a snippet of the first template method instantiation in a case study. The case study was scoped towards the function group *Course management* and related functionality, currently implemented in the Windows client platform. Only the software platform INTRANET was relevant in this case study, because the current functionality is designed for employees within the organization. For this example we have limited ourselves to the persona TEACHER, although other personas have been identified in the complete case study. We have selected the functionality VIEW PARTICIPANTS PER COURSE EVENT, originating from the entity COURSE EVENT. Given the scenario TEACHER ON INTRANET and the selected functionality, a MAPPING was assigned with a relatively high *Priority* of *0.9*. This is because in the group session, it became

obvious that during a course day, a teacher is particularly interested in the number of participants and their names and organizations, which helps a teacher to prepare for the course.

In the first case study, 135 sets of functionality were initially identified. 90 sets of functionality were never assigned a mapping, which means they are currently not relevant in the evolution of the software suite for this software platform. Of the remaining 45 sets of functionality, an average of the mapping priorities was calculated per functionality. The functionality, the scenarios and their mappings are presented in a matrix, sorted in a descending order by their average mapping priority.

The second case study was scoped towards functionality in the function group *Fixed assets management*. It initiated with a total of 79 sets of functionality, of which 56 sets were assigned a mapping and priority. This implies that 23 sets of functionality were excluded from the software evolution by discussing them in the group session.

VI. Related work

In requirements prioritization, different techniques exist, each with a different level of complexity. Examples are the composition of a *top ten*, *ranking* [39], *cumulative voting* [42], *priority groups* such as *MoSCoW* [48] and *requirements triage* [46], the *binary priority list* [41], and *Karl Wieger's prioritization matrix* [43]. The selection of the correct technique depends on the complexity and magnitude of the project, and the skills and competencies of the project manager or analyst.

The Actor Dependency (AD) model [49] analyzes the software processes to capture *why* a software process has been implemented by a software developing organization, rather than *how* it was implemented, or *what* the process was designed like. This creates a better understanding of the composition of software development processes and the motivations, intents and rationales behind them.

Generally, strategies to cope with legacy information systems can be subdivided into three categories: *redevelopment*, *wrapping* and *migration* [1]. The difference in impact on the current and new system give a good impression of the wide range of aspects to take into account in software evolution.

The context of legacy system reengineering can be seen from the perspective of *engineering*, *system*, *software*, *managerial*, *evolutionary* and *maintenance* [50]. Each perspective comes with challenges to be considered, to realize an effective approach towards reengineering.

The Chicken Little Methodology [51] is considered to be the most mature approach for the migration of a software product [52]. However, the approach makes extensive use of gateways, which increases the complexity of the software. Therefore, the Butterfly Methodology is a gateway-free approach to legacy system migration which reduces the risk of increasing complexity [52].

VII. Discussion

The research project's goal is to design a template method for software developing organizations. Since the role of a software product manager does not necessarily imply having in-depth knowledge about the product's source code and technical architecture, techniques that concern technical competencies such as the analysis of source code or implemented architecture, are omitted. Thus, the software product is analyzed from a functional perspective. However, this might not be conclusive, and a functional approach may produce more overhead and consume more resources compared to technical, potentially automated approaches that were left out in the research.

During the extraction of entities and functionality from the software product, functionality is not clustered, nor are relationships between functionality recorded. Should we have decided to do so, the process of mapping functionality on scenarios would have become too complex, due to a cascading assignment of priorities among functionality. This does not benefit the efficiency of the template method, as the mapping phase is not designed to consider such extensive dependencies.

The template method's deliverable, an instantiation of the concept REPORT, is not suitable to be considered a product roadmap. The method does not take into account the required and available resources for the implementation of each set of functionality during the mapping phase. The required resources are not exclusively dependent on the characteristics of the functionality itself, as the difficulty of implementing functionality can be different per software platform. However, it has been a conscious decision to exclude the consideration of resources, as it would increase the complexity of the mapping phase, making the method less efficient. The prioritized shortlist of requirements, delivered by the method as a report, can in turn be used as input for roadmap intelligence, as described in Section I and visualized in Figure 1.

The template method has been designed, instantiated and validated at a single case company, which produces an integrated ERP software product. It may be possible that validation at another case company, producing different software products, possibly even adhering to another development methodology, may result in different performance. Such validation is left open for future work.

VIII. Conclusion

This research presents the Software Functional Evolution Method (SFEM), a structured approach towards the evolution of a software product by mapping functionality between software platforms. The template method is designed for software product managers. The foundations are concerned with a functional, rather than a technical perspective on software analysis.

An initial template method is designed on the basis of a literature review and expert interviews. By means of instantiating the template method in case studies, the performance of the method is analyzed, which makes it possible to improve the method by designing method increments.

The method increments are captured and a categorization is applied to each increment. The categorization enables the analysis of the complete and incremental method engineering

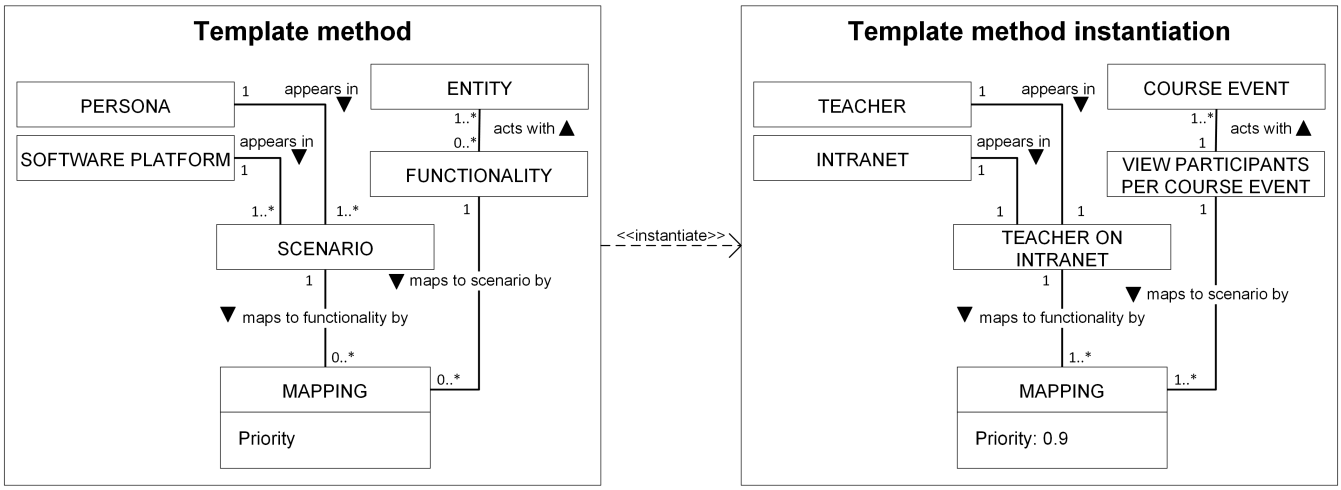


Fig. 6. Template method instantiation for Course management

process, which allows for reflection on the process by the researcher.

A. Future work

The two case studies in which the template method is instantiated are performed at one single software developing organization in the Netherlands, which produces an integrated ERP system for the Dutch marketplace. Future work validates and improves the template method by means of case studies at other software vendors, with different software products and different project requirements. Thus, a more quantitative approach towards the instantiation and validation of the template is subject to future research.

The research project is designed for software product managers, and thus technical approaches for the analysis of software products and their architecture are omitted. Future work explores the (semi-)automated analysis of software products, to make the process of extracting entities and functionality more efficient and conclusive.

In the mapping phase of the template method, opportunities exist for automated application and cascading of priorities, based on relationships between entities and functionality. However, this is not explored in this research project, as it is still unclear what effect cascading has on the outcome of an instantiation. Such dependencies would require a thoroughly tested algorithm which automatically cascades a mapping's priority based on properties of the relationship between entities or functionality.

ACKNOWLEDGMENTS

We would like to thank all interviewees who have participated in the case studies and group sessions for their knowledge and cooperation during the research project. Their contribution has been of great significance to the development of the template method.

REFERENCES

[1] J. Bisbal, D. Lawless, B. Wu, and J. Grimson, "Legacy information systems: Issues and directions," *Software, IEEE*, vol. 16, no. 5, pp. 103–111, 1999.

[2] J. T. Yee and S.-C. Oh, "Focusing on RFID, Interoperability, and Sustainability for Manufacturing, Logistics, and Supply Chain Management," in *Technology Integration to Business*. Springer, 2013, pp. 67–95.

[3] V. T. Rajlich and K. H. Bennett, "A staged model for the software life cycle," *Computer*, vol. 33, no. 7, pp. 66–71, 2000.

[4] C. Ebert, "The impacts of software product management," *Journal of Systems and Software*, vol. 80, no. 6, pp. 850–861, Jun. 2007.

[5] W. Bekkers, I. van de Weerd, M. Spruit, and S. Brinkkemper, "A Framework for Process Improvement in Software Product Management," in *Systems, Software and Services Process Improvement*. Springer Berlin Heidelberg, 2010, vol. 99, pp. 1–12.

[6] D. Moher, A. Liberati, J. Tetzlaff, and D. G. Altman, "Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement," *Annals of Internal Medicine*, vol. 151, no. 4, pp. 264–269, 2009.

[7] F. Harmsen, S. Brinkkemper, and H. Oei, *Situational Method Engineering for Information System Project Approaches*. University of Twente, Department of Computer Science, 1994, no. September.

[8] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools," *Information and software technology*, vol. 38, no. 4, pp. 275–280, 1996.

[9] I. van de Weerd and S. Brinkkemper, "Meta-Modeling for Situational Analysis and Design Methods," in *Handbook of research on modern systems analysis and design technologies and applications*. Information Science Reference, 2008, vol. 35, pp. 35–54.

[10] H. van der Schuur, "Process Improvement through Software Operation Knowledge: If the SOK Fits, Wear It!" *SIKS Dissertation Series*, vol. 2011, no. 43, 2011.

[11] R. K. Yin, *Case study research: Design and methods*. Sage, 2009.

[12] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Dec. 2009.

[13] International Software Product Management Association. (2014) Software Product Management Body of Knowledge (SPMBOK). [Online]. Available: http://ispma.org/spmbok/

[14] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[15] Object Management Group, "UML 2.0 superstructure specification," Technical Report ptc/04-10-02, Tech. Rep., 2004.

[16] "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Standard 610.12*, 1990.

[17] "Systems and software engineering – Software life cycle processes," *IEEE Standard 12207*, 2008.
[18] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.
[19] "IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document," *IEEE Standard 1362*, 1998.
[20] "IEEE Standard for Conceptual Modeling Language - Syntax and Semantics for IDEF1X97 (IDEFobject)," *IEEE Standard 1320.2*, 1998.
[21] A. Cooper, *The inmates are running the asylum: Why hightech products drive us crazy and how to restore the sanity*. Indianapolis, IN: SAMS, Macmillan Computer Publishing, 1999.
[22] "IEEE Standard for Adoption of ISO/IEC 26513:2009 Systems and Software Engineering–Requirements for Testers and Reviewers of Documentation," *IEEE Standard 26513*, 2010.
[23] "IEEE Standard for Information Technology–Systems Design–Software Design Descriptions," *IEEE Standard 1016*, 2009.
[24] "ISO/IEC/IEEE Systems and software engineering – Content of lifecycle information products (documentation)," *IEEE Standard 15289*, 2011.
[25] G. G. Chowdhury, "Natural language processing," *Annual Review of Information Science and Technology*, vol. 37, no. 1, pp. 51–89, 2003.
[26] R. Kazman, L. O'Brien, and C. Verhoef, "Architecture Reconstruction Guidelines, Third Edition," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. November, 2003.
[27] R. Kazman and S. J. Carrière, "Playing Detective: Reconstructing Software Architecture from Available Evidence," *Automated Software Engineering*, vol. 6, no. 2, pp. 107–138, 1999.
[28] L. O'Brien, C. Stoermer, and C. Verhoef, "Software Architecture Reconstruction: Practice Needs and Current Approaches," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. August, 2002.
[29] J. Martin, *Managing the data base environment*. Prentice Hall PTR, 1983.
[30] M. Stolze, P. Riand, M. Wallace, and T. Heath, "Agile Development of Workflow Applications with Interpreted Task Models," in *6th international conference on Task Models and Diagrams for User Interface Design*. Springer, 2007, pp. 2–14.
[31] C. G. Jung and A. E. Storr, *The essential Jung*. Princeton University Press, 1983.
[32] M. Aoyama, "Persona-and-scenario based requirements engineering for software embedded in digital consumer products," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, 2005, pp. 85–94.
[33] P. T. A. Junior and L. V. L. Filgueiras, "User modeling with personas," in *Proceedings of the 2005 Latin American conference on Human-computer interaction*. New York, New York, USA: ACM Press, 2005, pp. 277–282.
[34] A. Cooper, R. Reimann, and D. Cronin, *About Face 3: The Essentials of Interaction Design*. John Wiley & Sons, 2012.
[35] A. Gawer and M. A. Cusumano, "Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation," *Innovation: Management, Policy & Practice*, vol. 5, no. 1, pp. 91–94, 2003.
[36] J. Bosch, "From Software Product Lines to Software Ecosystems," in *Proceedings of the 13th International Software Product Line Conference*, no. Splc. Carnegie Mellon University, 2009, pp. 111–119.
[37] S. Mann, "Wearable computing: a first step toward personal imaging," *Computer*, vol. 30, no. 2, pp. 25–32, 1997.
[38] A. Gawer, *Platform dynamics and strategies: from products to services*. Cheltenham: Edward Elgar Publishing Limited, 2009.
[39] P. Berander and A. Andrews, "Requirements Prioritization," in *Engineering and Managing Software Requirements*. Springer Berlin Heidelberg, 2005, pp. 69–94.
[40] Z. Racheva, M. Daneva, and L. Buglione, "Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products," in *Second International Workshop on Software Product Management*, Barcelona, Catalunya, 2008, pp. 49–58.
[41] T. Bebensee, I. van de Weerd, and S. Brinkkemper, "Binary Priority List for Prioritizing Software Requirements," in *Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2010, pp. 67–78.
[42] D. Leffingwell and D. Widrig, *Managing software requirements: a unified approach*. Addison-Wesley Professional, 2000.
[43] K. Wiegers, "First things first: prioritizing requirements," *Software Development*, vol. 7, no. 9, pp. 48–53, 1999.
[44] I. Sommerville and P. Sawyer, *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.
[45] DSDM Consortium, *DSDM Atern Handbook*. DSDM Consortium, 2008. [Online]. Available: http://www.dsdm.org/content/10-moscow-prioritisation
[46] A. M. Davis, "The art of requirements triage," *Computer*, vol. 36, no. 3, pp. 42–49, 2003.
[47] I. van de Weerd, S. Brinkkemper, and J. Versendaal, "Concepts for Incremental Method Evolution: Empirical Exploration and Validation in Requirements Management," in *Advanced Information Systems Engineering SE - 33*, ser. Lecture Notes in Computer Science, J. Krogstie, A. Opdahl, and G. Sindre, Eds. Springer Berlin Heidelberg, 2007, vol. 4495, pp. 469–484.
[48] DSDM Consortium, *MoSCoW Prioritisation — DSDM Atern Handbook*. DSDM Consortium, 2008. [Online]. Available: http://www.dsdm.org/content/10-moscow-prioritisation
[49] E. S. Yu and J. Mylopoulos, "Understanding "why" in software process modelling, analysis, and design," in *Proceedings of the 16th international conference on Software engineering*. IEEE Computer Society Press, 1994, pp. 159–168.
[50] S. R. Tilley and D. Smith, "Perspectives on Legacy System Reengineering," 1995.
[51] M. L. Brodie and M. Stonebraker, *Migrating legacy systems: gateways, interfaces & the incremental approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995.
[52] B. Wu, D. Lawless, J. Bisbal, R. Richardson, J. Grimson, V. Wade, and D. O'Sullivan, "The butterfly methodology: A gateway-free approach for migrating legacy information systems," in *Third IEEE International Conference on Engineering of Complex Computer Systems*. IEEE, 1997, pp. 200–205.

Appendix B

Activity table

The activities of a Process-Deliverable Diagram (PDD) represent the process of the designed method. Table B.1 describes the activities of the Software Functionality Evolution Method. The activities of the method are presented on the left side of the PDD in Figure 4.1. The notation is based on the UML activity diagram (Object Management Group, 2004).

Main activity	Sub activity	Description
Project definition	Write project plan	The project's outline is defined, so that a common understanding of the project's goal and properties is set. This concept is also included in the REPORT. The property *Scope* of the concept limits the exploration of ENTITIES and FUNCTIONALITY in the project instantiation.
	Identify stakeholders	Each STAKEHOLDER of the project is identified and his/her *Role* in the project is noted. The person who instantiates the template method is also a STAKEHOLDER, as are STAKEHOLDERS who are only interested in the final REPORT. In Section 4.3.1, a Method Stakeholder Classification Matrix is proposed, which can help in the identification and labeling of the *Role* of the STAKEHOLDER.
	Define domain ontology	By analyzing the ENTITIES within the domain of discourse and how those ENTITIES are related, a DOMAIN ONTOLOGY can be defined (Gruber, 1993). The underlying descriptive models of the software product can assist in retrieving an accurate description of the DOMAIN ONTOLOGY. The DOMAIN ONTOLOGY lays a basis for the DATA MODEL.

Functionality identification	Identify entities within scope	Limited by the *Scope* of the PROJECT DEFINITION, the ENTITIES within the project instantiation's *Scope* are identified. The underlying descriptive models of the software product can help in the identification of ENTITIES.
	Identify functionality	Based on identified the ENTITIES and an analysis of the software product in scope, FUNCTIONALITY is identified and linked to the ENTITIES it corresponds with. Section 4.4.1 describes different methods on how to identify FUNCTIONALITY from a functional perspective. FUNCTIONALITY can be organized by a functionality classifier, as described in Section 4.4.2, which makes identification of all FUNCTIONALITY within the project scope easier and more efficient.
	Define entity relationships	By defining the relationships amongst ENTITIES, the concept DATA MODEL is instantiated. The relationships can be extracted from underlying descriptive models, and from descriptions of FUNCTIONALITY, by interpreting on which ENTITIES the FUNCTIONALITY acts.
Scenario creation	Define personas	Based on the actual users of the software product, PERSONAS are defined to represent them. A PERSONA can be both an actual as a potential user of the software product, which means that a PERSONA can also represent a user group from a new market segment which is about to be explored. The exploration of PERSONAS is, however, limited by the *Scope* and *Goal* of the PROJECT DEFINITION. In Section 4.5.1, different properties of a PERSONA are explored, which help in the instantiation of the concept.
	Define software platforms	After the PERSONAS in the project have been identified, an analysis of new SOFTWARE PLATFORMS can be made. As with the PERSONAS, the SOFTWARE PLATFORMS align with the PROJECT DEFINITION's *Scope* and *Goal*. Section 4.5.2 explains more about the analysis of SOFTWARE PLATFORMS from a functional perspective.

	Create scenarios	By combining use cases of PERSONAS on the defined SOFTWARE PLATFORMS, SCENARIOS are defined. It is possible that a PERSONA is not present on a SOFTWARE PLATFORM, which means this combination does not produce a SCENARIO.
Functionality mapping	Identify mapping candidates	By iterating over the instantiations of FUNCTIONALITY for each SCENARIO, an initial mapping is made which defines whether or not it would make sense to provision the FUNCTIONALITY for a given SCENARIO. If so, an instantiation of the concept MAPPING is made, with an empty *Priority* property. If the FUNCTIONALITY should not be provisioned for the given SCENARIO, no MAPPING is instantiated.
	Review mapping candidates	The reviewing of MAPPING candidates is performed in a group session with selected and relevant STAKEHOLDERS. The initial MAPPINGS are discussed and MAPPINGS can be added or removed. No *Priority* is yet assigned to the MAPPING during this activity.
	Prioritize mapping candidates	Parallel with the activity *Review mapping candidates*, the MAPPING candidates are reviewed by assigning a *Priority*. The activity is performed in a group session with selected and relevant STAKEHOLDERS. The complexity of this activity depends on the complexity of the project instantiation, the complexity of the software product and the depth of the analysis of FUNCTIONALITY. If the decision of a MAPPING is complex or not sufficiently obvious, the decision can be captured in a DESIGN RATIONALE. Different prioritization techniques, which can help in the calculation of the *Priority*, are described in Section 4.6.1.
Results reporting	Select stakeholders	Before a REPORT can be created, a set of STAKEHOLDERS has to be selected, for whom to report to. The interests of the STAKEHOLDERS determine the level of detail in the REPORT.

Create report	Tuned to the interests of the selected STAKEHOLDERS, a REPORT of the project instantiation's outcome is created. In Section 4.7, more information about the concept REPORT and categorizations is given.
Report to stakeholders	The created REPORT is presented to the selected STAKEHOLDERS, and a discussion is held to determine whether there is consensus about the REPORT and results being satisfying to all STAKEHOLDERS. If the STAKEHOLDERS can not reach consensus about the results in the REPORT, a new group session is initiated to review the MAPPINGS. In the case where no new iteration over the MAPPINGS has to be made, the project instantiation comes to an end.

Table B.1: Activity table of the SFEM

Appendix C

Concept table

The deliverables of a template method instantiation, represented in Table C.1, result from the execution of activities within an instantiated method. Formal definitions based on the IEEE Standards Definition Database[1] are included, with support of scientific literature and the results from the case study method instantiations.

The concepts of the template method are visualized on the right side of the Process-Deliverable Diagram in Figure 4.1. The notation of concepts is based on the UML class diagram (Object Management Group, 2004). All instantiations of concepts result from the performance of activities, except for the concept DESIGN RATIONALE, which is implicitly incorporated in the activities instantiating the concept MAPPING.

Concept	Description
PROJECT PLAN	The PROJECT PLAN is a document that describes the technical and management approach to be followed for a project. The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way that the project will be organized (IEEE Std. 610.12-1990). The PROJECT PLAN is included in the REPORT of the project's results, and in any other documentation that acts as a deliverable of the project. The concept's properties *Scope* and *Goal* play an important role in the further instantiation of the template method.
STAKEHOLDER	A STAKEHOLDER is an individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations (IEEE Std. 12207-2008). The STAKEHOLDER has a predefined *Role* in the project. At least one STAKEHOLDER is the project manager, who is the person with overall responsibility for the management and running of a project (ISO/IEC/IEEE 26512). A STAKEHOLDER's role can be organized as described in Section 4.3.1.
DOMAIN ONTOLOGY	An ontology describes the ENTITIES within the domain in discourse, and how these ENTITIES are interrelated (Gruber, 1993). The DOMAIN ONTOLOGY represents the domain in which the software product is designed to operate, the domain of discourse. It is composed of higher-level ENTITIES which are identified in the product's functional architecture. The DOMAIN ONTOLOGY lays the basis for the DATA MODEL.

[1] http://dictionary.ieee.org

ENTITY	In computer programming, an ENTITY is any item that can be named or denoted in a program. For example, a data item, program statement, or subprogram. (IEEE Std. 610.12-1990). The concept FUNCTIONALITY is applicable to ENTITIES, and the relationships between ENTITIES are represented in the DATA MODEL.
FUNCTIONALITY	FUNCTIONALITY concerns the capabilities of the various computational, user interface, input, output, data management, and other features provided by a product (IEEE Std. 1362-1998). Techniques to identify software FUNCTIONALITY have been described in Section 4.4.1. To organize the instances of FUNCTIONALITY, a classification can be applied, as is described in Section 4.4.2.
DATA MODEL	A DATA MODEL identifies the entities, domains (attributes), and relationships (associations) with other data and provides the conceptual view of the data and the relationships among data (IEEE Std. 1320.2-1998). The DATA MODEL serves as a basis for the remainder of a template method instantiation and can be included in a REPORT. A DATA MODEL can be represented by a UML class diagram (Booch et al., 1999), but a simple tree diagram can also suffice, depending on the complexity of the software product at hand.
PERSONA	PERSONAS are defined as representations of the actual users of a system, defined by the goals they aim to accomplish. Personas are hypothetical archetypes of actual users (Cooper, 1999). A PERSONA appears in at least one SCENARIO, which maps it to at least one SOFTWARE PLATFORM. More information about the defining of a PERSONA can be found in Section 4.5.1.
SOFTWARE PLATFORM	A platform is the combination of an operating system and hardware that makes up the operating environment in which a program runs (ISO/IEC 26513). Thus, a SOFTWARE PLATFORM defines the environment in which a software product is designed to operate. In Section 4.5.2, more information about the description of a SOFTWARE PLATFORM is given.
SCENARIO	The combination of possible and relevant appearances of PERSONAS on SOFTWARE PLATFORMS creates an instantiation of the concept SCENARIO. A SCENARIO is used to map FUNCTIONALITY by MAPPINGS and their *Priorities*.
MAPPING	A MAPPING is an assigned correspondence between two things that is represented as a set of ordered pairs (IEEE Std. 1320.2-1998). The MAPPING concept plays a central role in the REPORT of the results of a template method instantiation. A MAPPING is created by the combination of a SCENARIO with FUNCTIONALITY. If no mapping occurs at the SCENARIO, meaning that a certain FUNCTIONALITY is excluded from a SCENARIO, the MAPPING does not exist. The *Priority* of the MAPPING is determined based on the importance of a set of FUNCTIONALITY for a given SCENARIO. If a MAPPING has no *Priority* assigned, it is considered to be a candidate. The decision and rationale behind a MAPPING can be captured in a DESIGN RATIONALE.
DESIGN RATIONALE	A DESIGN RATIONALE is defined as information capturing the reasoning of the designer that led to the system as designed, including design options, trade-offs considered, decisions made, and the justifications of those decisions (IEEE Std. 1016-2009). It presents the arguments behind a MAPPING and its *Priority*. If the MAPPING changes due to an iteration after the *Results reporting* activities, another DESIGN RATIONALE can be assigned to the same MAPPING. More information about the concept DESIGN RATIONALE can be found in Section 4.6.2.

REPORT	A REPORT is an information item that describes the results of activities such as investigations, observations, assessments, or tests (ISO/IEC/IEEE 15289). The results of a template method instantiation are communicated to selected STAKEHOLDERS by a REPORT, which is designed to suit the STAKEHOLDER's interests. As the needs of the STAKEHOLDER, and thus the details of the REPORT, highly depend on the situation at hand, we define the REPORT as a closed complex concept. A REPORT incorporates at least one or more instantiations of the concepts PROJECT PLAN, FUNCTIONALITY, SCENARIO and MAPPING. Section 4.7 describes possible formats of a REPORT.

Table C.1: Concept table of the SFEM

Appendix D

Case study: Course management in AFAS InSite

Product	CRM
Functiegroep	Cursusmanagement
Platform	AFAS InSite
Persona	Employee (Dutch: Medewerker) Trainer (Dutch: Docent) Training manager (Dutch: Cursusbeheerder)

This section describes the execution of the first case study of our research. Its goal is to instantiate the designed method in a closed environment, analyze its performance and design method increments so that the method can evolve by improvement.

We begin by describing the applied method as it was designed before the execution of the case study (Section D.1). In Section D.2 we describe the instantiation that results from executing the method. The results of the analysis of the template method instantiation performance are further elaborated in Section 5.1.

D.1 Template method

This section explains the method as it was designed before the execution of this case study. The method's design is left untouched during the execution phase of the case study, so that its performance can be analyzed and method increments can be captured. These method increments serve as feedback for the method's design, so that it can be improved. This process is also depicted in Figure 2.3, where the activity *Project performance* serves *Requests for adaptations* to the activity *Assembly of method fragments*.

For this case study, the conceptual method in Figure D.1 has been applied.

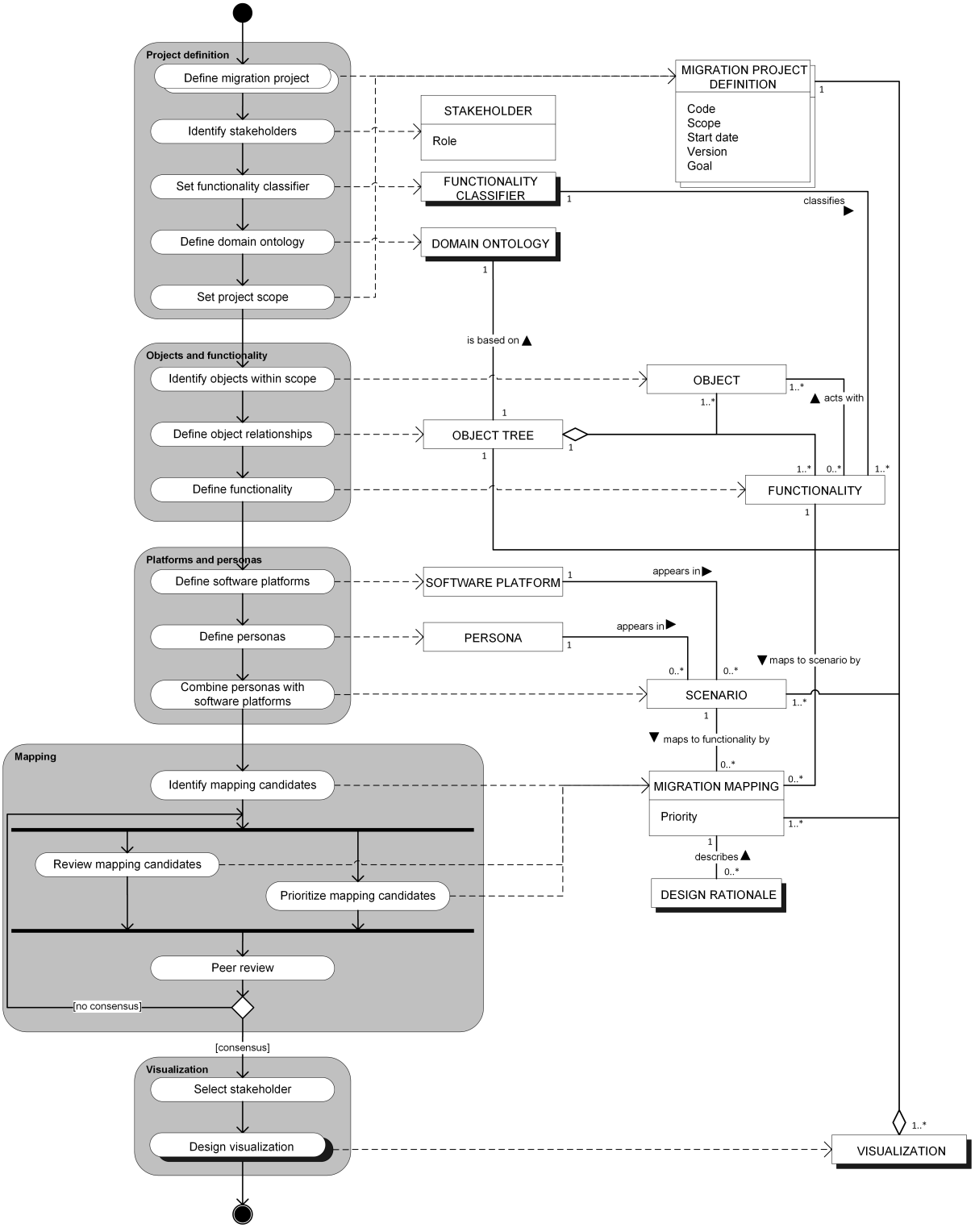


Figure D.1: Template method 1: Process-Deliverable Diagram

D.1.1 Activity table

The activities of the method are listed in Table D.1.

Main activity	Sub activity	Description
Project definition	Define migration project	The project's outline is defined, so that a common understanding of the project's goal and properties is made explicit.
	Identify stakeholders	Each STAKEHOLDER of the project is identified and his/her role in the project is noted. The project manager is also a STAKEHOLDER, as well as STAKEHOLDERS who are only interested in the final deliverable as VISUALIZATION.
	Set functionality classifier	Depending on the complexity of the software product at hand, a FUNCTIONALITY CLASSIFIER is set.
	Define domain ontology	By analyzing the entities within the domain of discourse and how those entities are related (Gruber, 1993), a DOMAIN ONTOLOGY is defined. The structure of the data model of the software product can assist in retrieving an accurate description of the DOMAIN ONTOLOGY.
	Set project scope	The scope of the project is set and saved in the PROJECT DEFINITION. The project scope depends on the goal of the project at hand.
Objects and functionality	Identify objects within scope	Limited by the scope in the PROJECT DEFINITION, the OBJECTS that are within the scope are identified. The software product's data model can assist in the retrieval of the OBJECTS.
	Define object relationships	The OBJECT TREE is formed by identifying relationships between the different OBJECTS. Relationships are based on the UML class diagram's relationship types (Booch et al., 1999), so that a common understanding of the meaning is maintained.

	Define functionality	Based on the OBJECTS in the OBJECT TREE, FUNCTIONALITY is identified and linked to the OBJECTS it corresponds with. FUNCTIONALITY is defined by the project's FUNCTIONALITY CLASSIFIER.
Platforms and personas	Define software platforms	The SOFTWARE PLATFORMS that are within the scope of the PROJECT DEFINITION are defined. They align with the goal of the project at hand.
	Define personas	For each SOFTWARE PLATFORM, the types of users that can work with the SOFTWARE PLATFORM are defined as PERSONAS.
	Combine personas with software platforms	By combining use cases of PERSONAS on the defined SOFTWARE PLATFORMS, SCENARIOS are defined.
Mapping	Identify mapping candidates	The identification of mapping candidates involves the definition of either absolute inclusion or exclusion of FUNCTIONALITY to given SCENARIOS. If FUNCTIONALITY is not expected in a given SCENARIO, no MIGRATION MAPPING is created and a void remains.
	Review mapping candidates	The reviewing of mapping candidates is performed in a group session with selected and relevant STAKEHOLDERS. The previously determined mapping candidates are again discussed and decided upon inclusion or exclusion. This decision is based on and adjusted by the decisions made over other mapping candidates. For example, the inclusion of one MIGRATION MAPPING could imply the exclusion of another mapping candidate, even though it has been included as mapping candidate earlier.

	Prioritize mapping candidates	Parallel with the previous activity, mapping candidates are prioritized, based on the importance of the MIGRATION MAPPING for the given SCENARIO. The complexity of the project at hand determines the the adhered method to calculate the priority for the MIGRATION MAPPING. If the decision of a MIGRATION MAPPING is complex or not sufficiently obvious, the decision can be captured in a DESIGN RATIONALE.
	Peer review	The MIGRATION MAPPING and optionally its DESIGN RATIONALE are discussed in a peer review session. This can be either group sessions or individual interviews, depending on the complexity and absence of sufficient domain knowledge. If no consensus is reached, the priority of the MIGRATION MAPPING is reconsidered.
Visualization	Select stakeholder	In order to design and perform the VISUALIZATION, the STAKEHOLDER for whom the VISUALIZATION is designed, is selected.
	Design visualization	Specialized for the selected STAKEHOLDER, a VISUALIZATION is designed and created.

Table D.1: Template method 1: Activity table

D.1.2 Concept table

Concept	Description
MIGRATION PROJECT DEFINITION	The PROJECT DEFINITION includes metadata about the project to be executed. It is included in the VISUALIZATION of the project's outcome, and in any other documentation that acts as a deliverable of the project.
STAKEHOLDER	A STAKEHOLDER is anyone who plays a significant role in the project. The STAKEHOLDER has a predefined role in the migration project. At least one STAKEHOLDER is the project's project manager.
FUNCTIONALITY CLASSIFIER	The FUNCTIONALITY CLASSIFIER helps to classify FUNCTIONALITY, so that a standard definition of FUNCTIONALITY can be maintained.
DOMAIN ONTOLOGY	An ontology describes the objects within the domain in discourse, and how these objects are interrelated (Gruber, 1993). The DOMAIN ONTOLOGY represents the domain in which the software product is designed to operate, the domain of discourse. It is composed of higher-lever entities which are identified in the product's functional architecture. The DOMAIN ONTOLOGY helps in the structuring of the OBJECT TREE.
OBJECT	An OBJECT is an entity in the software product in scope, to which FUNCTIONALITY has been applied. It is equal to an entity in the UML class diagram (Booch et al., 1999) and can have different types of relationships with other OBJECTS. The OBJECTS and their relationships are included in the OBJECT TREE.
OBJECT TREE	The OBJECT TREE contains the OBJECTS and their FUNCTIONALITY within the software product. It is based on the DOMAIN ONTOLOGY. The notation for the object tree is the UML class diagram (Booch et al., 1999).
FUNCTIONALITY	FUNCTIONALITY concerns the behavior the software product is designed to perform for the user. It acts with at least one OBJECT. The FUNCTIONALITY CLASSIFIER is used to set a standard definition for FUNCTIONALITY.
SOFTWARE PLATFORM	A SOFTWARE PLATFORM defines the environment in which a software product is designed to operate within. It can appear in SCENARIOS in order to be combined with a PERSONA.
PERSONA	Cooper (1999) defines PERSONAS as representations of the users of a system, defined by the goals they wish to accomplish. They are hypothetical archetypes of actual users.
SCENARIO	By combining possible appearances of PERSONAS to SOFTWARE PLATFORMS, we create SCENARIOS. A SCENARIO is used to map FUNCTIONALITY in the MIGRATION MAPPING. It plays a central role in the VISUALIZATION of the output of the project.

MIGRATION MAPPING	The MIGRATION MAPPING concept plays a central role in the VISUALIZATION of the output of the project. It is formed by the combination of SCENARIOS with FUNCTIONALITY. If no mapping occurs at the SCENARIO, meaning that a certain FUNCTIONALITY is excluded from a SCENARIO, the MIGRATION MAPPING does not exist (defined in the activity of identifying mapping candidates). The priority of the MIGRATION MAPPING is determined based on its importance in the migration project. The decision about the MIGRATION MAPPING can be captured in a DESIGN RATIONALE. If the MIGRATION MAPPING changes due to the group sessions or peer review, another DESIGN RATIONALE is assigned.
DESIGN RATIONALE	A DESIGN RATIONALE captures the decision making arguments for a MIGRATION MAPPING's priority. If the MIGRATION MAPPING changes due to the expert evaluation, another DESIGN RATIONALE is assigned, which implies that one MIGRATION MAPPING can have multiple DESIGN RATIONALES.
VISUALIZATION	The output of a migration project is communicated to selected STAKEHOLDERS in a VISUALIZATION, which is designed to suit the STAKEHOLDER's interests, for instance by means of the degree of details. Example VISUALIZATIONS include different technology roadmap types (purpose and format) (Phaal et al., 2004) or a spreadsheet with SCENARIOS as columns and FUNCTIONALITY as rows.

Table D.2: Template method 1: Concept table

D.2 Template method instantiation

As suggested by Van der Schuur et al. (2011), creating an instance of a method by instantiating activities and objects helps to explicate and analyze the actual performance of a method. This enabled analysis of the method's performance, which can in turn lead to improvement in terms of method increments.

This section describes the instantiation of the method's activities and concepts, as reported in Section D.1.

D.2.1 Activity instantiations

Define migration project

The case study's migration project takes place at AFAS Software, an ERP software vendor from Leusden, The Netherlands. The software suit comprises different products, such as ERP, CRM, HRM, logistics and finance. This case study is known by code **AFAS.Profit.CRM.001**. The project's scope is prematurely set to Cursusmanagement (training planning), a function group of the CRM product. The scope is addressed and expanded in a later activity of the method. The start date of the case study and thus the method instantiation is at **3 February 2014**. This is the **first version** of the case study at the training planning module. The goal of the project is to **compare the output of the migration project with already completed evolution of the product suite**.

The concept which acts as output of the activity *Define migration project* is the MIGRATION PROJECT DEFINITION. The result of the instantiation is captured in Section D.2.2.

Set project code The project code is written in alphanumeric characters, separated by dots. Any long textual string can be abbreviated in order to improve legibility. The project code consists of four sections. The first section defines the organization's name. The second section defines the software product suite which is concerned in the migration project. The third section defines the product in the software product suite which is addressed. The fourth and last section of the project code is an identifier which can be used to distinguish multiple multiple projects within the same product.

Format: Organization.Suite.Product.Identifier

Value: AFAS.Profit.CRM.001

Set project start date The project start date is set to the first day at which the migration project is initiated.

Format: DD-MM-YYYY

Value: 03-02-2014

Set project version The project version is defined by numeric characters, separated by dots. It consists of two sections. The first section defines the major updates with a numeric identifier. The second section defines the minor updates with a numeric identifier.

Format: Major.Minor

Value: 1.0

Describe project goal The migration project's goal is described in terms of the organization's and migration project's aim and vision.

> The goal of this migration project is to execute the project for the CRM product's Cursusmanagement (training planning) function group, and compare these results with the actual evolution of the product's functionality. The software product has already evolved its functionality of Cursusmanagement to new software platforms by means of pragmatic reasoning and project management. This provides sufficient means to compare the outcome of pragmatic evolution with structured evolution of software functionality. The software platforms to be migrated towards are AFAS InSite and AFAS OutSite, web-based software platforms which run in the user's browser. The personas include all personas which can be identified on the AFAS InSite and AFAS OutSite platforms. The required deliverable of the project is a software product roadmap for the functionality in the Cursusmanagement function group, with priorities defining which functionality to migrate, and in which order.

Identify stakeholders

The STAKEHOLDERS of this migration project are identified in collaboration with the product management team of AFAS Software. Each stakeholder must play a significant role in the contribution to the project, either in a direct or indirect way.

In order to classify the stakeholders of the project, we introduce a classification matrix that is based on the interaction with the method's deliverables, and the degree of participation in the method execution. Based on these two parameters, we introduce the following "stakeholder classification matrix":

		Degree of participation	
		Active	Passive
Degree of interaction	Direct	**Participant** - Project manager - Product manager - Domain expert	**Observer** - Project manager - Product manager - Board member
	Indirect	**Informer** - Domain expert - Customer - Partner	**Outsider** - Developer - Designer

Figure D.2: Method Stakeholder Classification Matrix

Participation The degree of *Participation* defines how active a stakeholder is within the execution of the project. Active stakeholders play a role during the length of the project execution, while passive stakeholders are only interested in a portion of the project, or only just its deliverables.

Interaction The degree of *Interaction* defines how direct a stakeholder acts with the deliverables of the project. A direct relationship implies that the stakeholder gets the results of the project directly through its deliverables, or even contributes to the creation of the deliverables. An indirect relationship implies that the stakeholder only sees results of the project in an edited, more narrow way, without being aware of the decisions or precise details.

Participant The stakeholder role *Participant* is an active stakeholder who directly interacts with the deliverables of the project. The stakeholder can be involved as the manager of the project, or someone who delivers direct input through means of expert validations or interactions based on questions of the project manager.

Observer The stakeholder role *Observer* is a passive stakeholder who directly interacts with the deliverables of the project. This could be someone who needs to approve or understand the output of the project, without it being edited by stakeholders other than those with the role Participant.

Informer The stakeholder role *Informer* is an active stakeholer who indirectly interacts with the deliverables of the project. This means that in order for the stakeholder to interact with the deliverables, a passthrough from a direct stakeholder is necessary. The Informer often acts as a provider of knowledge for the Participant stakeholder role.

Outsider The stakeholder role *Outsider* is a passive stakeholder who indirectly interacts with the deliverables of the project. The stakeholder plays no role in the decision making project and does not get to see the whole picture of the project's deliverables, but is rather provided with a modified view with an as-is status.

Based on the proposed classification and the stakeholders within the product development team of AFAS Software, the stakeholders, their roles and classification are identified and the concept STAKEHOLDER is instantiated in Section D.2.2.

Notice how the table of stakeholders does not list any Outsiders at this point of time. That is because there is no actual plan to actually develop the method's outcome yet. As mentioned before, the purpose of this migration project is only to compare the outcome of a structured approach with that of a pragmatic approach.

Set functionality classifier

In AFAS Profit, an authorization tool is present to control which users can perform specific functionality in the software product. The authorization tool allows an administrator to give access to user groups, known as "roles". In the AFAS Profit Windows product, the following rights can be provided to users:

- No access
- View only
- Maintain

However, we believe that the web-based software platforms allow for much more personas and functionality to be executed, which is why we wish to expand the classification beyond these three classes. The CRUD classification of operations is therefore a better, more detailed specification which suits the use of a wide range of functionality better. The abbreviation CRUD stands for *Create*, *Read*, *Update* and *Delete*, which can be seen as the fundamentals of any software operation. The article by Cooper et al. (2010) suggests the addition of the *Scan* operation, which allows for reading a number of records and defines the *Read* method as reading only a single record. We do not think this adds any extra value in this case study, as in most cases a single record can only be accessed through an aggregated view of multiple records.

To conclude, in this migration project the FUNCTIONALITY CLASSIFIER is the CRUD classification of operations, which is specified in the instantiated concept FUNCTIONALITY CLASSIFIER in Section D.2.2.

Define domain ontology

The AFAS Profit software product is a suite of multiple information system functions combined. For instance, the software product does not only include functionality that is common in an enterprise resource planning (ERP) application, but also includes a product for customer relationship management (CRM) and human resource management (HRM). These are marketed as different products with different licenses, while the software product is packaged in one application and runs from one architecture. The products are also integrated, which means that an entity of a person can simultaneously be a trainee (CRM), an employee of the organization (HRM), while ordering products from the organization's catalog (ERP).

Therefore, the highest level of entities in the ontology is the products within the AFAS Profit suite. A product is then subdivided into function groups, which serve a shared goal or theme. For instance, the CRM product has the function group *Training planning* (Cursusmanagement), which manages trainings provided by an organization to its clients. A level lower is are entities, which are objects which can be acted upon by functionality. For instance, the function group *Training planning* has objects which represent the trainings themselves, the corresponding occurrences as events, and trainees as attendees. In the hierarchy of the ontology, objects can have other objects as parents or children in relationships. An example is the relationships between *Training*, *Event*, *Session*, *Trainee* and *Organization/Person*.

The domain ontology for this migration project is represented in the instantiated concept DOMAIN ONTOLOGY in Section D.2.2.

Set project scope

The project scope has been prematurely set in this method instantiation's activity *Define migration project* in Section D.2.1. The scope is set to Cursusmanagement (training planning), which is at the DOMAIN ONTOLOGY level of Functiegroep. This means that all Gegevens and Functionaliteit of the Functiegroep are included.

The activity contributes to the instantiated concept MIGRATION PROJECT DEFINITION.

Identify objects within scope

The scope, as defined in the instantiated concept MIGRATION PROJECT DEFINITION, is set to the Functiegroep *Cursusmanagement*. Looking at the DOMAIN ONTOLOGY, the concept OBJECT is instantiated as *Gegeven* in this migration project. Because a Gegeven can have relationships with other Gegevens, which leads to the instantiation of the concept OBJECT TREE, we show all objects in this activity, and create relationships between them in the next activity.

We first describe the objects that are directly in the function group of Cursusmanagement. This excludes the objects that are included in the migration project because they are related or inherited by objects that are directly included in the function group. The data has been acquired by running the tool *Gegevensverzamelingen* in AFAS Profit, which returns a list of OBJECTS in the AFAS Profit database, and in which Product and Functiegroep they are included.

Because we do not look any further than the objects that are directly in the function group Cursusmanagement, we miss out on the OBJECTS and FUNCTIONALITY that are important for Cursusmanagement to work, yet are not directly linked. Therefore, the next activity (*Define object relationships*) goes deeper into the function group, identifying OBJECTS that are also indirectly linked to the function group Cursusmanagement. These OBJECTS are then included in the full view of the OBJECT TREE.

Define object relationships

The instantiated concept OBJECT can be related to one another by defining relationships, from which the concept OBJECT TREE is instantiated. The relationships have been defined by analyzing a full-access environment in AFAS Profit in which relationships between objects are represented by tabs in views in the user interface.

The resulting instantiation of the concept OBJECT TREE is represented in Section D.2.2. The UML class diagram notation has been omitted, because in this case study it has been determined that a simple descending tree visualization would serve the cause well enough.

As mentioned in the previous activity, this activity also reveals other OBJECTS that are included in the function group Cursusmanagement, but through and indirect relationship. This means the OBJECTS are related to OBJECTS that are directly included in Cursusmanagement.

In the OBJECT TREE, we have stopped to go deeper into the OBJECT relationships when we identified an OBJECT which is too complex, for instance by having too many relationships with other OBJECTS. This is the case at the OBJECT *Organisatie/persoon*. This OBJECT has too many relationships with other OBJECTS, yet out of scope in Cursusmanagement, and is therefore not further elaborated in the OBJECT TREE.

Define functionality

The concept FUNCTIONALITY has been instantiated by assistance of the instantiated concept FUNCTIONALITY CLASSIFIER, which is CRUD. As was described in Section D.2.1, the original classification of functionality which is used in AFAS Profit does not align well enough with the diversity of personas in the new software platforms.

The operators *Create*, *Read*, *Update* and *Delete* have been applied to the different OBJECTS that have been identified. The result was that most of the FUNCTIONALITY was based on the CRUD FUNCTIONALITY CLASSIFIER. However, the user interface exposed other functionality as well, which goes beyond the direct application of four CRUD operators to OBJECTS. For instance, we identified FUNCTIONALITY that implied both the *Update* of an OBJECT, while *Deleting* another.

An example is the FUNCTIONALITY with ID 42: *Organisatie/persoon samenvoegen met andere Organisatie/persoon*. This FUNCTIONALITY moves all data and relationships from Organisatie/persoon A to Organisatie/persoon B, and then removes Organisatie/persoon A.

Because the OBJECT TREE inherited OBJECTS that were not directly in the function group Cursusmanagement, we had to expand our research into other function groups in order to give a full overview of the functionality that is either directly or indirectly integrated in the migration project's scope. However, to avoid the migration project becoming too complex and large, we have limited the listing of FUNCTIONALITY to only those OBJECTS that are included in the function group Cursusmanagement, or play a significant role in the FUNCTIONALITY of the function group.

For instance, we have excluded the further elaboration of the FUNCTIONALITY that is applicable to the OBJECT Administratie, because the FUNCTIONALITY is not relevant to the function group Cursusmanagement. On the other hand, we did include FUNCTIONALITY on the OBJECT Organisatie/persoon, because this functionality is inherited by the OBJECTS Docent and Deelnemer (Cursist), which both are included in the function group Cursusmanagement.

In the identification and definition of FUNCTIONALITY, no source code or official data representations have been consulted. The FUNCTIONALITY has been defined by interacting with the user interface of the software product and interpreting this into the instantiation of the required concepts. This approach has been chosen, because the analysis of source code or official data representations would produce too much overhead, which does not serve a significant contribution to the migration project.

By listing all OBJECTS within the OBJECT TREE, it has been able to identify those OBJECTS within the migration project's scope which still miss their applicable functionality. A complete list of all OBJECTS that are either directly or indirectly inherited by functiegroep Cursusmanagement is shown in Table D.8.

Define software platforms

The SOFTWARE PLATFORM in scope of this migration project is called AFAS InSite. AFAS InSite is a web-based SOFTWARE PLATFORM which runs in a web browser. It does not require any additional third-party software in order to operate, such as Adobe Flash[1] or Microsoft Silverlight[2]. In order to enlarge compatibility with modern systems, the system relies on standard client-side features like Javascript and cookies, which are enabled by default in modern web browsers.

In order to use the web application, a user is required to login with his/her credentials. Apart from password recovery, no functionality is available to unauthorized users. This creates several advantages, such as:

- Performed actions can be linked to and tracked back to the authorized user
- A user can only access pages, views and data to which he/she has the required authorization
- Workflows can be assigned to users and followed up by means of tasks in AFAS InSite
- Personal and confidential information, such as upcoming meetings, declaration status and salaries, can be presented with discretion

The creation and implementation of web pages in AFAS InSite is performed from the web interface itself, without the necessity of complex configuration controls from the AFAS Profit Windows application. Administrators with the right level of authorization can use the Content Management System (CMS) functionality to create pages, include widgets, design menu structures, and more. Since changes need to be published before becoming available to all users, an administrator can design changes in a conceptual version, without interfering with the usability of active users. The online administration interface also allows the changing of the site's theme, without the necessity of touching a single line of code. This so-called *What You See Is What You Get* (WYSIWYG) editor makes it possible to manage the website's content and theme and structure, without prior knowledge of how to write code or understand how a website technically works.

The AFAS InSite application already hosts a set of functionality which is derived from the AFAS Profit Windows application, which hosts the original set of functionality of the software suite. Examples of function groups that have already been migrated include document management, workflow management, dossier items, and organizations/persons from the CRM product.

However, besides the migration of functionality that is already present in the AFAS Profit Windows application, the web-based software platform also enables the addition of new functionality. AFAS InSite puts great emphasis on Self Service. For instance, Employee Self Service (ESS) and Manager Self Service (MSS). These two concepts enable the vision of capturing information directly at its source (the employee/manager) and reduces the communication between employees via information carriers such as email or printed forms.

Define personas

This case study concerns a total of three PERSONAS. The lowest level is that of Employee (Dutch: Medewerker). An Employee is basically anyone in an organization with an active state of employment. The other two PERSONAS, Trainer (Dutch: Docent) and Training manager (Dutch: Cursusbeheerder), inherit the functionality and authorization of the basic PERSONA Employee, as they are also required to have an active state of employment.

[1] http://www.adobe.com/software/flash/about
[2] http://www.microsoft.com/silverlight

Since an Employee is any person with an active state of employment, we must proceed with caution when assigning FUNCTIONALITY to a SCENARIO which is based on the PERSONA Employee. The intention of an Employee in FUNCTIONALITY of Cursusmanagement is often informative, as they do not need to control or manage the processes that surround Trainings. This means he or she does not have the intention to add, change or remove data from a Training.

A Trainer inherits all the functionality and authorization of an Employee, and thus also has an active state of employment. In the case study at AFAS Software, a Trainer is most often a consultant, and sometimes a product manager. Trainings are provided by one Trainer at the time. Based on their expertise, a consultant is assigned to a Training on a regular basis. Since consultants are working with implementations, optimizations and knowledge transfer to customers on a daily basis, they have the right skills required to educate users and help them getting started with the AFAS Profit product.

A Training manager is not necessarily a Trainer, yet is an Employee. They manage the available Trainings, publish information about them in the customer portal, and manage subscriptions of Trainees. They perform administrative functions on Trainings either through AFAS Profit Windows or AFAS InSite. The main characteristics of the actions they perform are focused on planning, as the operational actions at the days of the Trainings themselves are handled by the Trainers.

Considering the access rights of an Employee, Trainer or Training manager, AFAS Profit offers two options of action at the termination of employment. First, when the employment is terminated, the account's authorization is automatically reduced to a level where the user can login, but does not have authority to access any page of InSite. Second, if the user's account is also locked (Dutch: geblokkeerd), the user is also not able to login, which locks him/her entirely out of the system. Based on this observation, we can conclude that any user in AFAS InSite is indeed an employee with an active state of employment.

Combine personas with software platforms

In this case study, we examine only one SOFTWARE PLATFORM and three PERSONA, which leads to a total of three combinations, instantiating the concept SCENARIO. The PERSONA Employee is very generic, and should therefore be treated carefully. The other two PERSONAS are more specific and can therefore be treated with more certainty. Because the PERSONAS Trainer and Training manager inherit functionality from the generic PERSONA Employee, any of the three PERSONAS have always got access to the AFAS InSite SOFTWARE PLATFORM of their organization, as they are in an active state of employment.

Identify mapping candidates

The activity of the identification of MIGRATION MAPPING candidates focuses on the question whether or not an instance of FUNCTIONALITY should be present in an instantiated SCENARIO. For this case study, that means whether or not a set of FUNCTIONALITY should be available for an Employee, Trainer or Training manager in AFAS InSite. This activity does not concern the assigned priority or rationale of a MIGRATION MAPPING, even though details about the decision may be captured in a DESIGN RATIONALE.

Looking at the SCENARIO of an Employee in AFAS InSite, we must proceed with caution when we decide about the MIGRATION MAPPING of FUNCTIONALITY. An Employee is a very generic description of a PERSONA, and its access rights are inherited by any logged in and employed user in AFAS InSite.

On the other hand, the SCENARIOS with the PERSONAS Trainer and Training manager are more specific and the activities and goals for the Trainer can be considered operational, while the Training manager performs more tactical or planning-centered activities.

The identification of MIGRATION MAPPING candidates is performed by one individual, Gerard Nijboer, who was identified as Project manager in the STAKEHOLDER matrix. The activity is considered to be a preparation for the activities followed by the *Identify mapping candidates* activity, which are *Review mapping candidates* and *Prioritize mapping candidates*. These parallel activities are performed in a group session, as interaction and discussion is required to prepare well enough for the *Peer review* activity.

As the selecting of candidates concerns an absolute definition of simply inclusion or exclusion of FUNCTIONALITY, the activity produces a list of MIGRATION MAPPING candidates which can be used as input for the next activities. However, it can also be concluded that any FUNCTIONALITY which does not have a MIGRATION MAPPING assigned, is considered excluded from the SCENARIO. This set of FUNCTIONALITY can then also be grouped together in a list of FUNCTIONALITY, which also serves as input for the parallel group session activities. The list gives a quick overview of the excluded FUNCTIONALITY, which improves the efficiency of the activity *Review mapping candidates*, because it provides not only a list of included FUNCTIONALITY, but also the excluded FUNCTIONALITY.

Review mapping candidates

In a group session with two product managers of AFAS Software, Henk van der Schuur and Mohamed Amri, this activity and the subsequent activity concerning the prioritization of MIGRATION MAPPINGS have been performed. This activity has been performed prior to the prioritization activity, even though we can say the activities run parallel as prioritization was discussed, yet not captured, during the first activity.

To start the group session, a short presentation of 5 minutes was held in order to create common ground on which to build the rest of the activity. This included some of the instantiated concepts of the activities in the method that have already been performed, such as the PERSONAS. Through means of open discussion, the identified mapping candidates have been discussed. This caused some movement in the eventual outcome of the MIGRATION MAPPING candidates. Even between both product managers, some positive discussion was held which caused some shifts in the MIGRATION MAPPINGS.

The changed MIGRATION MAPPINGS have directly been recorded into the database of the project. This made it possible to change MIGRATION MAPPINGS live on the screen, so that an actual representation of the project's progress was shown, which could also directly be validated by the members of the group session. In total, 32 changes were made to the MIGRATION MAPPINGS, which could either be a change from inclusion to exclusion or vice versa, or adding an additional MIGRATION MAPPING to the current list.

Prioritize mapping candidates

Due to a shortage of time, the session in which we had planned to review and prioritize the mapping candidates, was split into two sessions. In the second session, the mapping candidates of the first session were left as they resulted from the previous activity, though some FUNCTIONALITY was eventually left out, even though they were identified as mapping candidates. This was due to the fact that the priority was considered too low when assigning a priority.

In consultation with the two product managers who were collaborating in the case study, it was agreed that a priority for a given MIGRATION MAPPING should be any integer number from 0 to 5, where 0 would imply exclusion of the FUNCTIONALITY, 1 is the lowest priority, and 5 is the highest priority. However, during the execution of the project, it was recognized that we had not created an complete vision on the meaning of the numbers, as it left room for interpretation. For instance, a MIGRATION MAPPING with the highest priority does not imply that it is crucial for the migration project to be successful, nor does it imply whether it contributes to the minimum viable product, and if that was aimed for in the project. This is something we should have considered in the initiation phase of the migration project, where we needed to agree on the precise goal of the project.

In total, 97 mappings of the 135 sets of FUNCTIONALITY in the function group Cursusmanagement were updated. As with the previous session, this session was also recorded, which did not put too much pressure on the capturing of DESIGN RATIONALES. Therefore, only one DESIGN RATIONALE was created. Of the 106 changed DESIGN RATIONALES, 3 were initially identified as a mapping candidate, yet in this session assigned a priority of 0. Since there were 135 sets of functionality, and 3 PERSONAS and thus 3 SCENARIOS, a total of 405 sets of MIGRATION MAPPINGS could have been assigned. This means that 24 percent (97/405) of the potential MIGRATION MAPPINGS have resulted in an actual MIGRATION MAPPING. In total, 45 MIGRATION MAPPING candidates are accepted as a result of the activity.

The concept MIGRATION MAPPING has been instantiated in Section D.2.2. It shows only those combinations of FUNCTIONALITY with SCENARIOS that have been identified as potentials to be migrated to the SOFTWARE PLATFORM AFAS InSite. The priorities for the SCENARIOS Cursusbeheerder, Docent and Medewerker are displayed in the last three columns of the table. This is not the final VISUALIZATION, which is supposed to be tuned to selected STAKEHOLDERS.

Peer review

In order to start the peer review activity, a VISUALIZATION had to be designed to present the outcome of the case study so far. This included a list of FUNCTIONALITY, with the MIGRATION MAPPINGS in three columns. To support the integrity of the deliverable we wish to present, we have added multiple columns. One column sums the three priorities of the MIGRATION MAPPINGS per SCENARIO together, which implies a maximum of 15 points. The next column counts the number of MIGRATION MAPPINGS present for the FUNCTIONALITY that is covered in the row. The third added column shows the average priority for the FUNCTIONALITY, depending on the number of MIGRATION MAPPINGS.

We acknowledge that the resulting average is skewed when there is only one MIGRATION MAPPING present, compared to a more diverse set of FUNCTIONALITY with more MIGRATION MAPPINGS. Another way to calculate the average is taking the sum of the MIGRATION MAPPING priorities and dividing this by the total number of identified SCENARIOS, which is 3 in this case study. However, we have discussed the preferred calculation of the average with the peers, and we concluded to take the actual number of applicable SCENARIOS for a set of FUNCTIONALITY to calculate the average with.

Select stakeholder

For this migration project, the STAKEHOLDERS have already been defined in Section D.2.2. Based on this table of STAKEHOLDERS and their roles, we design a visualization for Henk van der Schuur (product manager) and Mohamed Amri (manager product management). The other STAKEHOLDERS are not included in the presentation of the deliverables, since this case study does not concern them at this moment of time.

Design visualization

For the VISUALIZATION of the case study for the selected STAKEHOLDERS, we have reused the VISUALIZATION we designed in the activity *Peer review*. By exporting the results to Excel, we have been able to add conditional formatting. The formatting colored the values in the last six columns in the color green. The intensity of the color is dependent on the value, which means a cell with value 5 is 100% green, and a cell with value 1 is left white. The rows in the VISUALIZATION have been sorted on the final column, in descending order. The next ordering takes place on the column *Gegeven* and *Functionaliteit*. This was decided in cooperation with the STAKEHOLDERS in the *Peer review* activity.

Since the number of STAKEHOLDERS for this case study was limited, we have decided to only create one VISUALIZATION, represented in Figure D.4.

As described in the Process-Deliverable Diagram and the concept table of the template method, the VISUALIZATION also includes a summary of the MIGRATION PROJECT DEFINITION. This helps to identify the case study for which the VISUALIZATION was designed.

D.2.2 Concept instantiations

MIGRATION PROJECT DEFINITION

MIGRATION PROJECT DEFINITION	
Code	AFAS.Profit.CRM.001
Scope	Cursusmanagement
Start date	3 February 2014
Version	1.0
Goal	Migration to AFAS InSite

Table D.3: MIGRATION PROJECT DEFINITION

STAKEHOLDER

Name	Role	Classification
Henk van der Schuur	Product manager	Participant
Gerard Nijboer	Project manager	Participant
Mohamed Amri	Manager product management	Informer
Bas van der Veldt	Chief executive officer	Observer
Dennis van Velzen	Chief product development	Observer

Table D.4: Stakeholders

		PARTICIPATION	
		Active	Passive
INTERACTION	Direct	**Participant** Gerard Nijboer Henk van der Schuur	**Observer** Bas van der Veldt Dennis van Velzen
	Indirect	**Informer** Mohamed Amri	**Outsider**

Table D.5: Instantiated stakeholder classification matrix

FUNCTIONALITY CLASSIFIER

Create: Creates a single record in the database

Read: Reads one or more records from the database

Update: Updates a single record from the database, adding or replacing properties (Cooper et al., 2010)

Delete: Deletes a single record from the database

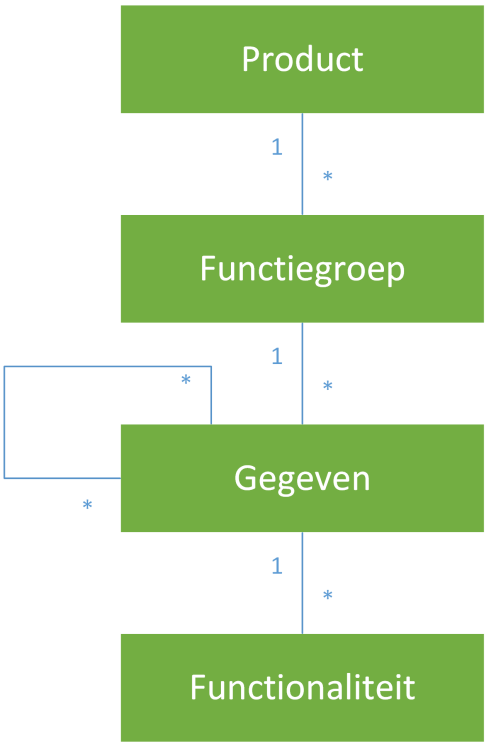


Figure D.3: DOMAIN ONTOLOGY

OBJECT

ID	Object
152	Bezettingsoverzicht
877	Cursus
17	Cursussessie
21	Cursusvoorkennis
16	Dagdeel
20	Deelnemer (Cursist)
15	Docent
19	Evenement
878	Presentie
884	Rol
885	Werkgebied

Table D.6: OBJECTS directly within Cursusmanagement

OBJECT TREE

Gegeven	**Functiegroep**
— Bezettingsoverzicht	Cursusmanagement
— Cursus	Cursusmanagement
—— Administratie	Administratie
—— Afbeelding	
—— Artikelgroep	Item
—— Btw-tariefgroep	
——— Land	Inrichting
—— Cursusvoorkennis	Cursusmanagement
—— Docent	Cursusmanagement
——— Organisatie/persoon	Organisatie/persoon

—— Dossieritem	Dossier
—— Eenheid	
—— Evenement	Cursusmanagement
——— Administratie	Administratie
——— Cursussessie	Cursusmanagement
———— Dagdeel	Cursusmanagement
———— Docent	Cursusmanagement
————— Organisatie/persoon	Organisatie/persoon
———— Locatie	Inrichting
——— Dagdeel	Cursusmanagement
——— Deelnemer (Cursist)	Cursusmanagement
———— Organisatie/persoon	Organisatie/persoon
———— Presentie	Cursusmanagement
——— Docent	Cursusmanagement
———— Organisatie/persoon	Organisatie/persoon
——— Dossieritem	Dossier
——— Locatie	Inrichting
——— Project	Project
——— Projectfase	Project
—— Extra barcode	Item
—— Kostprijs	Prijs/korting
—— Land	Inrichting
—— Locatie	Inrichting
—— Prijsgroep	Prijs/korting
—— Rol	Cursusmanagement
—— Verkoopfactuurregel	Facturen
—— Verkooporderregel	Verkoopproces
—— Verkoopprijs	Prijs/korting

—— Werkgebied	Cursusmanagement
— Cursussessie	Cursusmanagement
—— Dagdeel	Cursusmanagement
—— Docent	Cursusmanagement
——— Organisatie/persoon	Organisatie/persoon
—— Locatie	Inrichting
— Cursusvoorkennis	Cursusmanagement
— Dagdeel	Cursusmanagement
— Deelnemer (Cursist)	Cursusmanagement
—— Organisatie/persoon	Organisatie/persoon
—— Presentie	Cursusmanagement
— Docent	Cursusmanagement
—— Organisatie/persoon	Organisatie/persoon
— Evenement	Cursusmanagement
—— Administratie	Administratie
—— Cursussessie	Cursusmanagement
——— Dagdeel	Cursusmanagement
——— Docent	Cursusmanagement
———— Organisatie/persoon	Organisatie/persoon
——— Locatie	Inrichting
—— Dagdeel	Cursusmanagement
—— Deelnemer (Cursist)	Cursusmanagement
——— Organisatie/persoon	Organisatie/persoon
——— Presentie	Cursusmanagement
—— Docent	Cursusmanagement
——— Organisatie/persoon	Organisatie/persoon
—— Dossieritem	Dossier
—— Locatie	Inrichting

—— Project	Project
—— Projectfase	Project
— Presentie	Cursusmanagement
— Rol	Cursusmanagement
— Werkgebied	Cursusmanagement

Table D.7: Object tree

ID	Gegeven	Functiegroep
15	Docent	Cursusmanagement
16	Dagdeel	Cursusmanagement
17	Cursussessie	Cursusmanagement
19	Evenement	Cursusmanagement
20	Deelnemer (Cursist)	Cursusmanagement
21	Cursusvoorkennis	Cursusmanagement
39	Land	Inrichting
71	Organisatie/persoon	Organisatie/persoon
99	Dossieritem	Dossier
147	Afbeelding	
152	Bezettingsoverzicht	Cursusmanagement
264	Btw-tariefgroep	
593	Eenheid	
628	Verkoopfactuurregel	Facturen
640	Artikelgroep	Item
654	Extra barcode	Item
656	Kostprijs	Prijs/korting
659	Prijsgroep	Prijs/korting
661	Verkoopprijs	Prijs/korting

680	Verkooporderregel	Verkoopproces
827	Project	Project
830	Projectfase	Project
877	Cursus	Cursusmanagement
878	Presentie	Cursusmanagement
882	Administratie	Administratie
883	Locatie	Inrichting
884	Rol	Cursusmanagement
885	Werkgebied	Cursusmanagement

Table D.8: Inherited objects in Cursusmanagement

FUNCTIONALITY

Gegeven	CRUD	Functionaliteit
DAGDEEL	C	Dagdeel toevoegen
	R	Dagdeel weergeven
	U	Dagdeel bewerken
	D	Dagdeel verwijderen
CURSUSSESSIE	C	Cursussessie aan cursusevenement toevoegen
	R	Cursussessie van cursusevenement weergeven
	U	Cursussessie van cursusevenement bewerken
	D	Cursussessie van cursusevenement verwijderen
EVENEMENT	C	Evenement voor cursus toevoegen
	R	Evenement voor cursus weergeven
	U	Evenement voor cursus bewerken
	D	Evenement voor cursus verwijderen
DEELNEMER (CURSIST)	C	Deelnemer aan evenement toevoegen
	R	Deelnemers voor evenement weergeven
	U	Deelnemer voor evenement bewerken
	D	Deelnemer voor evenement verwijderen
	U	Deelnemer voor evenement verplaatsen
	C	Deelnemer voor evenement kopiëren
	R	Deelnemer per evenement weergeven
CURSUSVOORKENNIS	C	Cursusvoorkennis toevoegen aan cursus
	R	Cursusvoorkennis van cursus weergeven
	U	Cursusvoorkennis van cursus bewerken

Gegeven	CRUD	Functionaliteit
	D	Cursusvoorkennis van cursus verwijderen
LAND	R	Land weergeven
	U	Land bewerken
ORGANISATIE/PERSOON	C	Organisatie/persoon toevoegen
	R	Organisatie/persoon weergeven
	U	Organisatie/persoon bewerken
	D	Organisatie/persoon verwijderen
	U	Wijzigen persoon-/organisatiecode
	UD	Organisatie/persoon samenvoegen met andere Organisatie/persoon
	U	Omzetten van organisatie naar persoon en vice versa
	R	Afdrukken stamkaart
	C	Samenvoegregel toevoegen
	U	Samenvoegregel uitvoeren
	D	Samenvoegregel verwijderen
DOSSIERITEM	C	Dossieritem toevoegen
	R	Dossieritem weergeven
	U	Dossieritem bewerken
	D	Dossieritem verwijderen
	D	Dossieritem collectief verwijderen
	C	Subdossier toevoegen aan dossieritem
	D	Subdossier van dossieritem verwijderen
	R	Dossieritem afdrukken
	C	Insturen dossieritem
	C	Dossieritem op agendaplanning zetten
	C	E-mail dossieritem
	R	Bestemmingen van dossieritem weergeven
	R	Aanleiding van dossieritem weergeven
	R	Gebruikers per dossieritem weergeven
	R	Workflowhistorie van dossieritem weergeven
	R	Reacties van dossieritem weergeven
	C	Bijlage aan dossieritem toevoegen
	R	Bijlage van dossieritem weergeven
	U	Bijlage van dossieritem bewerken
	D	Bijlage van dossieritem verwijderen
	U	Bestemming van dossieritem bewerken
	U	Status “Afgehandeld” van dossieritem opheffen
	U	Dossieritem collectief bewerken
	U	Dossier archiveren
AFBEELDING	C	Afbeeldingen importeren
	C	Afbeelding toevoegen
	D	Afbeelding verwijderen

Gegeven	CRUD	Functionaliteit
	R	Afbeelding weergeven
	U	Afbeelding bewerken
	R	Afbeelding opslaan
	R	Afbeelding kopiëren
	CU	Afbeelding plakken
	U	Toelichting van afbeelding bewerken
	R	Afbeelding tonen in groot formaat
BEZETTINGSOVERZICHT	R	Bezettingsoverzicht cursussen weergeven
	R	Bezettingsoverzicht locaties weergeven
BTW-TARIEFGROEP	C	Btw-tariefgroep toevoegen
	R	Btw-tariefgroep weergeven
	U	Btw-tariefgroep bewerken
	D	Btw-tariefgroep verwijderen
EENHEID	R	Eenheid weergeven
VERKOOPFACTUURREGEL	R	Verkoopfactuurregel weergeven
ARTIKELGROEP	R	Artikelgroep weergeven
EXTRA BARCODE	R	Extra barcode weergeven
KOSTPRIJS	R	Kostprijs weergeven
PRIJSGROEP	R	Prijsgroep weergeven
VERKOOPPRIJS	R	Verkoopprijs weergeven
VERKOOPORDERREGEL	R	Verkooporderregel weergeven
PROJECT	R	Project weergeven
PROJECTFASE	R	Projectfase weergeven
CURSUS	C	Cursus toevoegen
	R	Cursus weergeven
	U	Cursus bewerken
	D	Cursus verwijderen
	U	Wijzigen cursuscode
PRESENTIE	R	Presentie van deelnemers weergeven
	U	Presentie van deelnemers bewerken
ADMINISTRATIE	R	Administratie openen
	CR	Administratie toevoegen
	D	Administratie verwijderen
	R	Administratie eigenschappen weergeven
	U	Administratie eigenschappen bewerken
	R	Informatie over Profit weergeven
	R	Reservekopie maken
	CU	Reservekopie terugzetten
LOCATIE	C	Locatie toevoegen
	R	Locatie weergeven
	U	Locatie bewerken

Gegeven	CRUD	Functionaliteit
	D	Locatie verwijderen
ROL	U	Rol van doelgroep bewerken
	C	Rol van doelgroep toevoegen
	R	Rol van doelgroep weergeven
	D	Rol van doelgroep verwijderen
WERKGEBIED	U	Werkgebied van doelgroep bewerken
	C	Werkgebied van doelgroep toevoegen
	R	Werkgebied van doelgroep weergeven
	D	Werkgebied van doelgroep verwijderen

Table D.9: Functionality of objects in Cursusmanagement

SOFTWARE PLATFORM

AFAS InSite

Functional context

- Online Content Management System allows for website administration without required knowledge of web programming
- Personal login inherits authorization and personal data and configuration
- Creating documents, pages and other content provides a *What You See Is What You Get* (WIYSIWYG) editor
- Operates in any modern web browser with cookies and Javascript enabled, and is thus platform-independent
- Changes to the website's administration can be designed in a conceptual environment, and be published in a later stage
- Strives for the implementation of AFAS' vision on Self Service (ESS, MSS)
- Users can login anywhere, at any time, independent of location, operating system or browser
- Cockpits give live insight into the performance of business units

Functional constraints

- Requires authorization for any functionality to be executed
- Requires a working internet connection in order to function
- Does not adjust to the display resolution of mobile devices
- User interface and human-computer interaction is different from the original Windows-interface, which might feel less natural

PERSONA

Employee An Employee (Dutch: Medewerker) is any person with an active state of employment at the case company. Any Employee has access to the SOFTWARE PLATFORM AFAS InSite with their personal credentials and profile. Since the organization does not only employ people who work with the AFAS Profit software, but, for instance, also employees for cleaning and the restaurant, we can not assume that any Employee has a technical background, or is familiar with the broad range of functionality in AFAS Profit. Therefore, it is hard to make any concrete statements about the competencies, knowledge, goals, needs, personal profile or constraints of an Employee.

Trainer A Trainer (Dutch: Docent) is often a consultant or a product manager of the case company, who is also an Employee, so in an active state of employment. The Trainer also inherits functionality and access rights from the generic Employee PERSONA. The reason why consultants provide trainings is because they have the necessary skills to educate users and to transfer their knowledge about how to implement and optimize the system. The competencies and knowledge of a Trainer are often focused on a specific product within the AFAS Profit software suite. However, a Trainer is always supposed to have sufficient knowledge about other products within the suite, as they are interlinked into one integrated software product. A consultant or product manager has commonly a degree in higher education, yet it is not required that they have completed a curriculum that is focused on software. A constraint of a Trainer is that none of the Trainers are dedicated Trainers. This means that they do not put their daily effort in practicing and preparing for trainings. However, since they work with implementations and optimizations of the software product and their focus product on a daily basis, and they most often give trainings on only their specific focus product, the fact that they do not extensively prepare for trainings, is not an issue.

Training manager A Training manager (Dutch: Cursusbeheerder) is an Employee with an active state of employment, and thus also inherits properties from the PERSONA Employee. The Training manager makes sure that all the information about trainings is correct and available for the customers, so that they can subscribe for an event. At the case company, this task is accommodated by the business unit entitle Customer operations. A Training manager does not necessarily need to have in-depth knowledge about the functioning of the software product, yet they must know what subjects are dealt with in a certain training. Their competencies focus on communication, as they need to make sure all the necessary data about trainings is available for customers and partners. Their goals are to sell trainings to customers by providing clear information about the trainings, and keeping the trainees satisfied. They are constrained by the fact that they are often not a Trainer themselves, which might cause them to lack insight in the daily activities of Trainers and their trainees.

SCENARIO

SCENARIO	**Employee in AFAS InSite**
PERSONA	Employee
SOFTWARE PLATFORM	AFAS InSite

SCENARIO	**Trainer in AFAS InSite**
PERSONA	Trainer
SOFTWARE PLATFORM	AFAS InSite

SCENARIO	**Training manager in AFAS InSite**
PERSONA	Training manager
SOFTWARE PLATFORM	AFAS InSite

Product	Functiegroep	Gegeven	Functionaliteit	Cursusbeheerder	Docent	Medewerker
Algemeen		Afbeelding	Afbeelding bewerken	5	5	5
Algemeen		Afbeelding	Afbeelding toevoegen	5	5	5
Algemeen		Afbeelding	Afbeelding tonen in groot formaat	5	5	5
Algemeen		Afbeelding	Afbeelding verwijderen	5	5	5
Algemeen		Afbeelding	Afbeelding weergeven	5	5	5
Algemeen	Inrichting	Locatie	Locatie weergeven	5	5	5
CRM	Cursusmanagement	Bezettingsoverzicht	Bezettingsoverzicht cursussen weergeven	5	5	2
CRM	Cursusmanagement	Bezettingsoverzicht	Bezettingsoverzicht locaties weergeven	5	5	2
CRM	Cursusmanagement	Cursus	Cursus bewerken	5		
CRM	Cursusmanagement	Cursus	Cursus toevoegen	5		
CRM	Cursusmanagement	Cursus	Cursus verwijderen	5		
CRM	Cursusmanagement	Cursus	Cursus weergeven	5	5	5
CRM	Cursusmanagement	Cursus	Wijzigen cursuscode	2		
CRM	Cursusmanagement	Cursussessie	Cursussessie aan cursusevenement toevoegen	5	2	
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement bewerken	5	2	
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement verwijderen	5		
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement weergeven	5	5	5
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis toevoegen aan cursus	2		
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus bewerken	2		
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus verwijderen	2		
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus weergeven	2	2	2
CRM	Cursusmanagement	Dagdeel	Dagdeel weergeven	2		
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer aan evenement toevoegen	5	2	
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer per evenement weergeven	5	5	2
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer voor evenement bewerken	5		
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer voor evenement verwijderen	5		
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemers voor evenement weergeven	5	5	2
CRM	Cursusmanagement	Evenement	Evenement voor cursus bewerken	5		
CRM	Cursusmanagement	Evenement	Evenement voor cursus toevoegen	5		
CRM	Cursusmanagement	Evenement	Evenement voor cursus verwijderen	5		
CRM	Cursusmanagement	Evenement	Evenement voor cursus weergeven	5	5	5
CRM	Cursusmanagement	Presentie	Presentie van deelnemers bewerken	4	4	1
CRM	Cursusmanagement	Presentie	Presentie van deelnemers weergeven	4	4	1
CRM	Dossier	Dossieritem	Bijlage aan dossieritem toevoegen	5	5	3
CRM	Dossier	Dossieritem	Bijlage van dossieritem weergeven	5	5	5

Product	Functiegroep	Gegeven	Functionaliteit	Cursusbeheerder	Docent	Medewerker
CRM	Dossier	Dossieritem	Dossieritem bewerken	5	5	3
CRM	Dossier	Dossieritem	Dossieritem toevoegen	5	5	3
CRM	Dossier	Dossieritem	Dossieritem verwijderen	5		
CRM	Dossier	Dossieritem	Dossieritem weergeven	5	5	5
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon bewerken	5	2	3
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon toevoegen	5	2	0
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon weergeven	5	5	5
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs bewerken	5		
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs toevoegen	5		
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs weergeven	5	1	1

VISUALIZATION

Product	Functiegroep	Gegeven	Functionaliteit	Cursusbeheerder	Docent	Medewerker	Sum	Count	Average
Algemeen		Afbeelding	Afbeelding toevoegen	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding verwijderen	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding weergeven	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding bewerken	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding tonen in groot formaat	5	5	5	15	3	5,0
CRM	Cursusmanagement	Cursus	Cursus weergeven	5	5	5	15	3	5,0
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement weergeven	5	5	5	15	3	5,0
CRM	Dossier	Dossieritem	Dossieritem weergeven	5	5	5	15	3	5,0
CRM	Dossier	Dossieritem	Bijlage van dossieritem weergeven	5	5	5	15	3	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus weergeven	5	5	5	15	3	5,0
Algemeen	Inrichting	Locatie	Locatie weergeven	5	5	5	15	3	5,0
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon weergeven	5	5	5	15	3	5,0
CRM	Cursusmanagement	Cursus	Cursus toevoegen	5			5	1	5,0
CRM	Cursusmanagement	Cursus	Cursus bewerken	5			5	1	5,0
CRM	Cursusmanagement	Cursus	Cursus verwijderen	5			5	1	5,0
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement verwijderen	5			5	1	5,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer voor evenement bewerken	5			5	1	5,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer voor evenement verwijderen	5			5	1	5,0
CRM	Dossier	Dossieritem	Dossieritem verwijderen	5			5	1	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus toevoegen	5			5	1	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus bewerken	5			5	1	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus verwijderen	5			5	1	5,0
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs toevoegen	5			5	1	5,0
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs bewerken	5			5	1	5,0
CRM	Dossier	Dossieritem	Dossieritem toevoegen	5	5	3	13	3	4,3
CRM	Dossier	Dossieritem	Dossieritem bewerken	5	5	3	13	3	4,3
CRM	Dossier	Dossieritem	Bijlage aan dossieritem toevoegen	5	5	3	13	3	4,3
CRM	Cursusmanagement	Bezettingsoverzicht	Bezettingsoverzicht cursussen weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Bezettingsoverzicht	Bezettingsoverzicht locaties weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemers voor evenement weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer per evenement weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Cursussessie	Cursussessie aan cursusevenement toevoegen	5	2		7	2	3,5
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement bewerken	5	2		7	2	3,5
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer aan evenement toevoegen	5	2		7	2	3,5
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon toevoegen	5	2		7	2	3,5

Product	Functiegroep	Gegeven	Functionaliteit	Cursusbeheerder	Docent	Medewerker	Sum	Count	Average
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon bewerken	5	2	3	10	3	3,3
CRM	Cursusmanagement	Presentie	Presentie van deelnemers weergeven	4	4	1	9	3	3,0
CRM	Cursusmanagement	Presentie	Presentie van deelnemers bewerken	4	4	1	9	3	3,0
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs weergeven	5	1	1	7	3	2,3
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus weergeven	2	2	2	6	3	2,0
CRM	Cursusmanagement	Cursus	Wijzigen cursuscode	2			2	1	2,0
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis toevoegen aan cursus	2			2	1	2,0
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus bewerken	2			2	1	2,0
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus verwijderen	2			2	1	2,0
CRM	Cursusmanagement	Dagdeel	Dagdeel weergeven	2			2	1	2,0

MIGRATION PROJECT DEFINITION	
Code	AFAS.PROFIT.CRM.001
Scope	Cursusmanagement
Start date	3 February 2014
Version	1.0
Goal	Migration to AFAS InSite

product	functiegroep	gegeven	functionaliteit	beheerder	docent	medewerker	SUM	COUNT	AVG
Algemeen		Afbeelding	Afbeelding toevoegen	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding verwijderen	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding weergeven	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding bewerken	5	5	5	15	3	5,0
Algemeen		Afbeelding	Afbeelding tonen in groot formaat	5	5	5	15	3	5,0
CRM	Cursusmanagement	Cursus	Cursus weergeven	5	5	5	15	3	5,0
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement weergeven	5	5	5	15	3	5,0
CRM	Dossier	Dossieritem	Dossieritem weergeven	5	5	5	15	3	5,0
CRM	Dossier	Dossieritem	Bijlage van dossieritem weergeven	5	5	5	15	3	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus weergeven	5	5	5	15	3	5,0
Algemeen	Inrichting	Locatie	Locatie weergeven	5	5	5	15	3	5,0
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon weergeven	5	5	5	15	3	5,0
CRM	Cursusmanagement	Cursus	Cursus toevoegen	5			5	1	5,0
CRM	Cursusmanagement	Cursus	Cursus bewerken	5			5	1	5,0
CRM	Cursusmanagement	Cursus	Cursus verwijderen	5			5	1	5,0
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement verwijderen	5			5	1	5,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer voor evenement bewerken	5			5	1	5,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer voor evenement verwijderen	5			5	1	5,0
CRM	Dossier	Dossieritem	Dossieritem verwijderen	5			5	1	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus toevoegen	5			5	1	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus bewerken	5			5	1	5,0
CRM	Cursusmanagement	Evenement	Evenement voor cursus verwijderen	5			5	1	5,0
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs toevoegen	5			5	1	5,0
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs bewerken	5			5	1	5,0
CRM	Dossier	Dossieritem	Dossieritem toevoegen	5	5	3	13	3	4,3
CRM	Dossier	Dossieritem	Dossieritem bewerken	5	5	3	13	3	4,3
CRM	Dossier	Dossieritem	Bijlage aan dossieritem toevoegen	5	5	3	13	3	4,3
CRM	Cursusmanagement	Bezettingsoverzicht	Bezettingsoverzicht cursussen weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Bezettingsoverzicht	Bezettingsoverzicht locaties weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemers voor evenement weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer per evenement weergeven	5	5	2	12	3	4,0
CRM	Cursusmanagement	Cursussessie	Cursussessie aan cursusevenement toevoegen	5	2		7	2	3,5
CRM	Cursusmanagement	Cursussessie	Cursussessie van cursusevenement bewerken	5	2		7	2	3,5
CRM	Cursusmanagement	Deelnemer (Cursist)	Deelnemer aan evenement toevoegen	5	2		7	2	3,5
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon toevoegen	5	2		7	2	3,5
CRM	Organisatie/persoon	Organisatie/persoon	Organisatie/persoon bewerken	5	2	3	10	3	3,3
CRM	Cursusmanagement	Presentie	Presentie van deelnemers weergeven	4	4	1	9	3	3,0
CRM	Cursusmanagement	Presentie	Presentie van deelnemers bewerken	4	4	1	9	3	3,0
Logistiek	Prijs/korting	Verkoopprijs	Verkoopprijs weergeven	5	1	1	7	3	2,3
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus weergeven	2	2	2	6	3	2,0
CRM	Cursusmanagement	Cursus	Wijzigen cursuscode	2			2	1	2,0
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis toevoegen aan cursus	2			2	1	2,0
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus bewerken	2			2	1	2,0
CRM	Cursusmanagement	Cursusvoorkennis	Cursusvoorkennis van cursus verwijderen	2			2	1	2,0
CRM	Cursusmanagement	Dagdeel	Dagdeel weergeven	2			2	1	2,0

Figure D.4: VISUALIZATION

Appendix E

Case study: Fixed assets in AFAS InSite

Product	Financieel
Function group	Vaste activa
Persona	Employee Facility manager ICT manager Financial controller Chief financial officer
Platform	AFAS InSite

E.1 Template method

This section explains the method as it was designed before the instantiation of this case study. The method's design is frozen during the execution phase of the case study, so that its performance can be analyzed and method increments can be captured. These method increments serve as feedback for the method's design, so that it can be improved. This process is also depicted in Figure 2.3, where the activity *Project performance* serves *Requests for adaptations* to the activity *Assembly of method fragments*.

For this case study, the conceptual method in Figure E.1 has been applied.

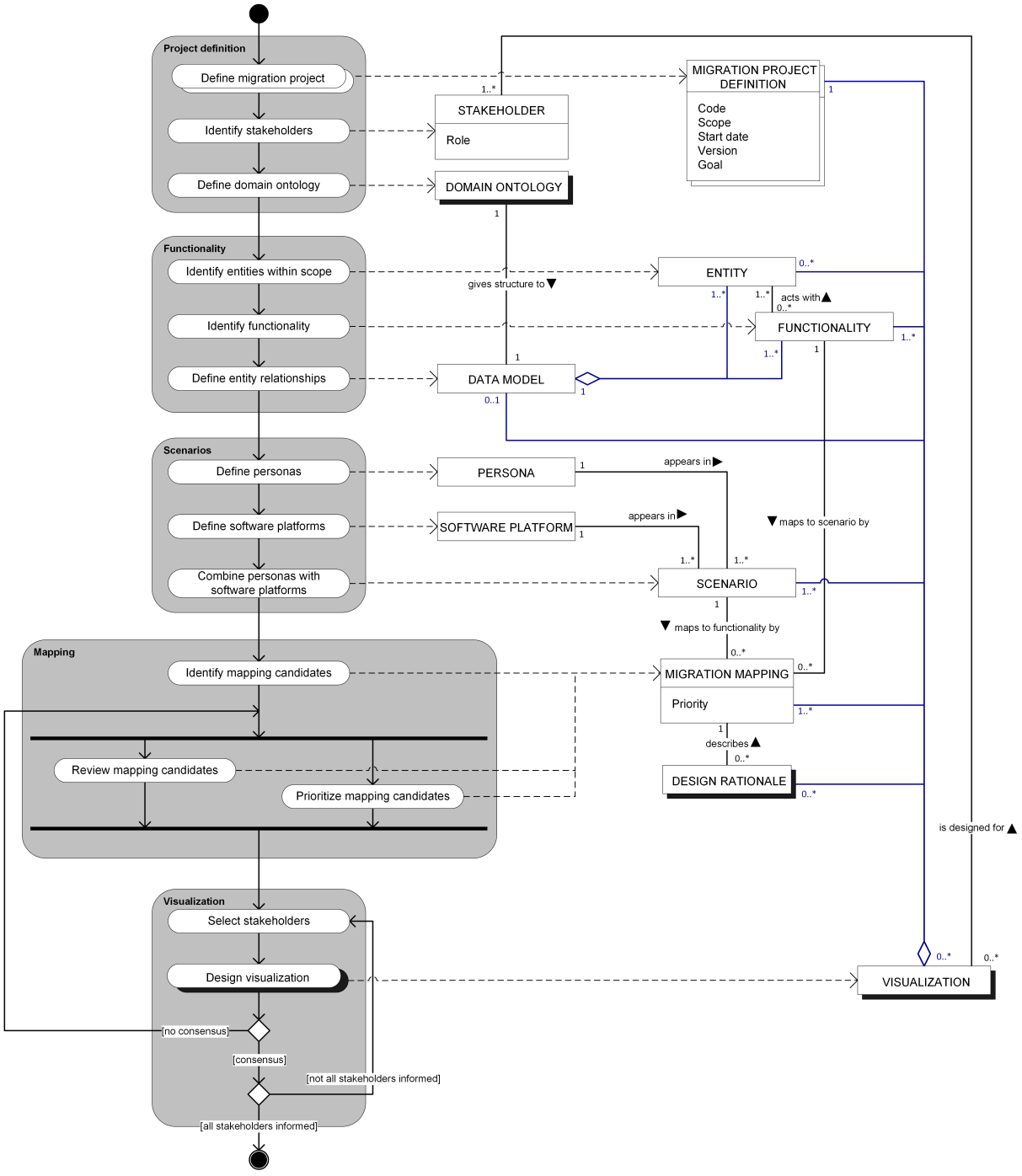


Figure E.1: Template method 2: Process-Deliverable Diagram

E.1.1 Activity table

The activities of the method are listed in Table E.1.

Main activity	Sub activity	Description
Project definition	Define migration project	The project's outline is defined, so that a common understanding of the project's goal and properties is set. This concept is also included in the VISUALIZATION. The property *Scope* of the concept limits the exploration of ENTITIES and FUNCTIONALITY in the migration project.
	Identify stakeholders	Each STAKEHOLDER of the project is identified and his/her *Role* in the project is noted. The person who instantiates the template method is also a STAKEHOLDER, as well as STAKEHOLDERS who are only interested in the final deliverable as VISUALIZATION. In Section 4.3.1, a Method Stakeholder Classification Matrix is proposed, which can help in the identification of the *Role* of the STAKEHOLDER.
	Define domain ontology	By analyzing the ENTITIES within the domain of discourse and how those ENTITIES are related, a DOMAIN ONTOLOGY can be defined (Gruber, 1993). The underlying descriptive models of the software product can assist in retrieving an accurate description of the DOMAIN ONTOLOGY.
Functionality	Identify entities within scope	Limited by the *Scope* in the MIGRATION PROJECT DEFINITION, the ENTITIES within the migration project's *Scope* are identified. The underlying descriptive models of the software product can help in the identification of ENTITIES.
	Identify functionality	Based on identified the ENTITIES and an analysis of the software product in scope, FUNCTIONALITY is identified and linked to the ENTITIES it corresponds with. Section 4.4.1 describes different methods on how to identify FUNCTIONALITY from a functional perspective. FUNCTIONALITY can be organized by a functionality classifier, as described in Section 4.4.2, which makes identification of all FUNCTIONALITY within the project scope easier.

	Define entity relationships	By defining the relationships amongst ENTITIES, the concept DATA MODEL is instantiated. The relationships can be extracted from underlying descriptive models, and from descriptions of FUNCTIONALITY, by interpreting on which ENTITIES the FUNCTIONALITY acts.
Scenarios	Define personas	Based on the actual users of the software product, PERSONAS are defined to represent them. A PERSONA can be both an actual as a potential user of the software product, which means that a PERSONA can also represent a user group from a new market segment which is about to be explored. The exploration of PERSONAS is, however, limited by the *Scope* and *Goal* of the MIGRATION PROJECT DEFINITION. More about the defining of PERSONAS is explained in Section 4.5.1.
	Define software platforms	After the PERSONAS have been identified, an analysis can be made of the SOFTWARE PLATFORMS. As with the PERSONAS, the SOFTWARE PLATFORMS align with the MIGRATION PROJECT DEFINITION's *Scope* and *Goal*. Section 4.5.2 explains more about the analysis of SOFTWARE PLATFORMS from a functional perspective.
	Combine personas with software platforms	By combining use cases of PERSONAS on the defined SOFTWARE PLATFORMS, SCENARIOS are defined.
Mapping	Identify mapping candidates	By iterating over the instantiations of FUNCTIONALITY for each SCENARIO, an initial decision is made which defines whether or not it would make sense to migrate the FUNCTIONALITY for the SCENARIO. If so, an instantiation of the concept MIGRATION MAPPING is made, with an empty *Priority* property. If the FUNCTIONALITY should not be migrated for the given SCENARIO, no MIGRATION MAPPING is instantiated.

	Review mapping candidates	The reviewing of MIGRATION MAPPING candidates is performed in a group session with selected and relevant STAKEHOLDERS. The initial decisions on MIGRATION MAPPINGS are discussed and MIGRATION MAPPINGS can be added or removed. No *Priority* is yet assigned to the MIGRATION MAPPING.
	Prioritize mapping candidates	Parallel with the activity *Review mapping candidates*, the MIGRATION MAPPING candidates are reviewed by assigning a *Priority*. The activity is performed in a group session with selected and relevant STAKEHOLDERS. The complexity of this activity depends on the complexity of the migration project, the complexity of the software product and the depth of the functional analysis of the FUNCTIONALITY. If the decision of a MIGRATION MAPPING is complex or not sufficiently obvious, the decision can be captured in a DESIGN RATIONALE. More information about the prioritization of MIGRATION MAPPINGS is described in Section 4.6.1.
Visualization	Select stakeholders	In order to design and perform the VISUALIZATION, a selection of STAKEHOLDERS for whom the VISUALIZATION is to be designed, is made.
	Design visualization	Tuned to the requirements of the selected STAKEHOLDERS, a VISUALIZATION of the migration project's outcome is designed. If the STAKEHOLDERS can not reach consensus about the deliverable of the migration project, a new group session is initiated to review the MIGRATION MAPPINGS. If not all required STAKEHOLDERS are informed yet, another iteration of the main activity *Visualization* is instantiated. More information about the designing of VISUALIZATIONS can be found in Section 4.7.

Table E.1: Template method 2: Activity table

E.1.2 Concept table

Concept	Description
MIGRATION PROJECT DEFINITION	The MIGRATION PROJECT DEFINITION includes metadata about the project to be executed. It is included in the VISUALIZATION of the project's outcome, and in any other documentation that acts as a deliverable of the migration project. The concept has some properties that play an important role in the further instantiation of the template method, such as *Scope* and *Goal*.
STAKEHOLDER	A STAKEHOLDER is anyone who plays a significant role in the project. The STAKEHOLDER has a predefined *Role* in the migration project. At least one STAKEHOLDER is the project's project manager, who is the individual who instantiates the template method and is responsible for the coordination. A STAKEHOLDER's role can be organized as described in Section 4.3.1.
DOMAIN ONTOLOGY	An ontology describes the ENTITIES within the domain in discourse, and how these ENTITIES are interrelated (Gruber, 1993). The DOMAIN ONTOLOGY represents the domain in which the software product is designed to operate, the domain of discourse. It is composed of higher-level ENTITIES which are identified in the product's functional architecture. The DOMAIN ONTOLOGY helps in the structuring of the DATA MODEL.
ENTITY	An ENTITY represents an object in the software product in scope, to which FUNCTIONALITY can been applied. It is equal to an ENTITY in the UML class diagram (Booch et al., 1999) and can have different types of relationships with other ENTITIES. The ENTITIES and their relationships are included in the DATA MODEL.
FUNCTIONALITY	FUNCTIONALITY concerns the behavior for which the software product has been designed to perform, on behalf of the user. An instance of FUNCTIONALITY acts with at least one ENTITY. To organize the instances of FUNCTIONALITY, a classification can be applied, as is described in Section 4.4.2. Techniques to identify software FUNCTIONALITY from a functional perspective has been described in Section 4.4.1.
DATA MODEL	The DATA MODEL is a formal representation of the ENTITIES, their FUNCTIONALITY, and how they are interconnected. The DATA MODEL serves as a basis for the rest of the template method instantiation and can be included in a VISUALIZATION. A DATA MODEL can be represented by a UML class diagram (Booch et al., 1999), but a simple tree diagram can also suffice, depending on the complexity of the software product and migration project.
PERSONA	PERSONAS are defined as representations of the actual users of a system, defined by the goals they aim to accomplish. They are hypothetical archetypes of actual users (Cooper, 1999). A PERSONA appears in at least one SCENARIO, which maps it to at least one SOFTWARE PLATFORM. More information about the defining of a PERSONA can be found in Section 4.5.1.

SOFTWARE PLATFORM	A SOFTWARE PLATFORM defines the environment in which a software product is designed to operate. It appears in at least one SCENARIO as is thereby combined with a PERSONA. Defining SOFTWARE PLATFORMS is based on the PERSONAS that have been identified, and their current or expected platforms they (wish to) use. In Section 4.5.2, more information about the describing of a SOFTWARE PLATFORM is given.
SCENARIO	By combining possible and relevant appearances of PERSONAS with SOFTWARE PLATFORMS, we create SCENARIOS. A SCENARIO is used to map FUNCTIONALITY in the MIGRATION MAPPING. It plays a central role in the VISUALIZATION of the output of the project.
MIGRATION MAPPING	The MIGRATION MAPPING concept plays a central role in the VISUALIZATION of the output of the project. It is formed by the combination of SCENARIOS with FUNCTIONALITY. If no mapping occurs at the SCENARIO, meaning that a certain FUNCTIONALITY is excluded from a SCENARIO, the MIGRATION MAPPING does not exist (defined in the activities *Identify mapping candidates* and *Review mapping candidates*). The *Priority* of the MIGRATION MAPPING is determined based on its importance in the migration project. If a MIGRATION MAPPING has no *Priority* assigned, it is considered to be a candidate. The decision and rationale behind the MIGRATION MAPPING can be captured in a DESIGN RATIONALE. If the MIGRATION MAPPING changes due to the group sessions or peer review, another DESIGN RATIONALE can be assigned.
DESIGN RATIONALE	A DESIGN RATIONALE captures the decision making arguments for a MIGRATION MAPPING's presence and *Priority*. If the MIGRATION MAPPING changes due to the activities in the *Mapping* main activity, another DESIGN RATIONALE can be assigned, which implies that one MIGRATION MAPPING can have multiple DESIGN RATIONALES. More information about the concept DESIGN RATIONALE can be found in Section 4.6.2.
VISUALIZATION	The output of a migration project is communicated to selected STAKEHOLDERS in a VISUALIZATION, which is designed to suit the STAKEHOLDER's interests, for instance by means of the degree of details. Example VISUALIZATIONS include different technology roadmap types (purpose and format) (Phaal et al., 2004) or a spreadsheet with SCENARIOS as columns and FUNCTIONALITY as rows. As the needs of the STAKEHOLDER, and thus the details of the VISUALIZATION, highly depend on the situation at hand, we define the VISUALIZATION as a closed concept. A VISUALIZATION incorporates at least one or more instantiations of the concepts MIGRATION PROJECT DEFINITION, FUNCTIONALITY, SCENARIO and MIGRATION MAPPING. Section 4.7 describes possible instantiations of the concept VISUALIZATION.

Table E.2: Template method 2: Concept table

E.2 Template method instantiation

DECISION POINT:
We use Dutch terms for entities that are represented in the sotware product. This makes it easier to link the descriptions with the actual entities and user interfaces of the software products, and improves consistency. We do, however introduce an English translation for each term the first time we mention it in the case study. Dutch terms are formatted in italics.

E.2.1 Activity instantiations

Define migration project

The case study is executed at AFAS Software, an ERP software vendor from Leusden, the Netherlands. The case study aims to migrate software functionality that is associated with *Vaste activa* (fixed assets) to any of the two web platforms of the AFAS Profit software suite, AFAS InSite and AFAS OutSite. The function group Vaste activa is part of the *Financieel* product (Finance) of AFAS Profit. We initialize the case study at **Monday 7 April 2014**. Since this is the **first version** of an assessment of the migration of functionality to a next-generation software platform for the function group *Vaste activa*, we assign the following code to this case study: **AFAS.Profit.ACT.001**.

The **scope** of this case study is aimed at functionality that is related to the entities within the function group *Vaste activa*. However, we do not limit ourselves to only those entities that are directly integrated in the function group. Other entities can be integrated with the entities directly in the function group, which means they need to be assessed in order to give a complete overview of the entities and functionality. Because the AFAS Profit software suite is one integrated software product, many direct relationships exist between entities in different function groups.

For the case study, the **goal** is set to make an assessment of whether the functionality of the function group *Vaste activa* should be migrated to the next-generation software platforms of the AFAS Profit software suite, which are the web platforms AFAS InSite and AFAS OutSite. The outcome of the case study can be used to determine whether or not it is worth the effort to migrate *Vaste activa* to the web platforms. The case study is an interesting assessment for the product management department of AFAS Software, because no product manager has explicitly considered the migration of the function group, just yet. This means that no prejudgments about the outcome can be made, and the outcome is highly dependent on the instantiation and performance of the Software Functionality Evolution Method.

This activity produces an instantiation of the concept MIGRATION PROJECT DEFINITION, which is projected in Table E.3.

Identify stakeholders

The STAKEHOLDERS of this case study are identified by means of consultation with the manager of the department product management. Furthermore, we have consulted formal design documents of the software product to identify authors who have participated in the design of functionality in the function group *Vaste activa*. Also, we include any STAKEHOLDERS who are addressed in research for the case study in a later stadium of the method instantiation, for instance when more in-depth knowledge needs to be gathered because the current STAKEHOLDERS can not provide sufficient information about the question.

We have again used the Method Stakeholder Classification Matrix, as described in Section 4.3.1, to assign a *Role* to the identified STAKEHOLDERS. We describe the *Role* of the STAKEHOLDERS in more detail in the paragraphs below.

Gerard Nijboer [Participant] Project manager, who instantiates the template method in the case study. He is responsible for the coordination of the project and monitoring its performance. The performance acts as feedback for method increments, which improve the template method.

Henk van der Schuur [Participant—Informer] Product manager of the *Financieel* product of AFAS Profit. He provides information and domain knowledge where necessary in the case study, such that the project manager can continue the method instantiation and case study. The domain knowledge apply to both the software product, as well as the industry at which it is applicable.

Mohamed Amri [Observer—Informer] Manager product management, who has a background in taxes and accountancy. He is primarily interested in the outcome of the method instantiation, but can also act as an *Informer* when other *Informers* are not available.

Jan Grijzen [Informer] Manager product development, who is responsible for the development of the *Financieel* product. He was involved in the initial design and development of *Vaste activa* in AFAS Profit, which makes him a valuable asset as an *Informer*.

Define domain ontology

The AFAS Profit software suite is an integrated software product that comprises different business software types. The following business software products can be identified within the single software product, in random order:

- General
- Customer Relationship Management (CRM)
- Logistics
- Projects
- Finance
- Taxes
- Human Resource Management (HRM)
- Payroll
- Subscriptions
- Reporting

The software suite originates as a Windows client, which connects to the AFAS Profit server. Currently, the software suite has been expanded to also run as a web and mobile application on different software platforms. The following software platforms are available:

AFAS Profit Windows Windows client application

AFAS InSite Intranet web application

AFAS OutSite Portal web application

AFAS Pocket iOS and Android smartphone application

For the continuation of this case study, when we refer to a “product”, we refer to a business software product within the AFAS Profit software suite. Each product has one or multiple function groups. A function group is a cohesion of entities and their functionality, which are specific for the execution of a certain task in the software product. Some entities may not have a function group assigned, as they are too generic or too specific. An entity can be related to other entities, which creates relationships and dependencies between different entities. Functionality, on the lowest level of the DOMAIN ONTOLOGY we define, is related to one or more entities.

The instantiated concept DOMAIN ONTOLOGY is visualized in Figure E.4.

Identify entities within scope

DECISION POINT:
The entity *Dagboek* is included, as it comes with the process of adding a *Vaste activa* through an *Inkoopfactuur*.

DECISION POINT:
When we speak of an instantiated concept, representing an entity in the software product, we transform the text into uppercase, as such: VASTE ACTIVA. When we speak of the entity in general, without the necessity of including its context in the project instantiation, we do not transform the text.

The *Scope* of the MIGRATION PROJECT DEFINITION is limited to those entities directly related to *Vaste activa*. However, during the identification of entities within the project *Scope*, we have included entities that are not directly related to the main entity VASTE ACTIVA. For instance, the entity INKOOPRELATIE (supplier) is included in order to be able to add new VASTE ACTIVA through an INKOOPFACTUUR (supplier invoice). The entity INKOOPRELATIE inherits the properties and relationships of the entity ORGANISATIE/PERSOON, which means we need to include this entity as well. However, we limit ourselves in the level of depth in the relationships when the entity loses its relevance in the *Goal* of this migration project.

We have excluded the entities TYPE ACTIEF, TYPE SUBSIDIE and LOCATIE. At first, these ENTITIES appeared to be actual ENTITIES in the DATA MODEL. However, since these ENTITIES and their values are controlled via the VRIJ TABEL, we do not include these ENTITIES and their FUNCTIONALITY separately.

Identify functionality

DECISION POINT:
During the identification of FUNCTIONALITY, we consulted with the STAKEHOLDER Henk van der Schuur to determine the relevance of FUNCTIONALITY for the following ENTITIES:

- ***Crediteur***
  Read only, as we only need to be able to select the already present *Crediteuren* in the *Administratie*. The maintaining of *Crediteuren* is not in the scope of this migration project.
- ***Dagboek***
  We open the *Dagboek* in order to be able to add an *Inkoopfactuur*.
- ***Grootboekrekening***
  The *Grootboekrekening* is necessary to add a *Vaste activa* through an *Inkoopfactuur*.
- ***Inkoopfactuur***
  An *Inkoopfactuur* can be used to add new *Vaste activa*.
- ***Inkoopfactuurregel***
  This ENTITY is at a too high level of detail, and is too far beyond the scope of this migration project.
- ***Medewerker***
  We can assign *Vaste activa* to the ENTITY *Medewerker*.
- ***Type actief***
  This is not an ENTITY which the customer can adjust in the software product.
- ***Type subsidie***
  This is an example of an ENTITY which can be maintained in the *Vrije tabellen*.

Based on the discussion with the STAKEHOLDER, we have determined to exclude the ENTITIES *Inkoopfactuurregel* and *Type actief* and their FUNCTIONALITY in the DATA MODEL.

In order to make the assigning of MIGRATION MAPPING more efficient, we have added descriptions to the FUNCTIONALITY where relevant and applicable. This helps to understand the precise behavior of the FUNCTIONALITY and ensures a common ground to rationalize upon.

The identification of FUNCTIONALITY in this case study has focused more on the guidance by functional processes instead of single items of FUNCTIONALITY. By thinking in processes, we enable ourselves to identify all relevant FUNCTIONALITY, instead of only the FUNCTIONALITY that is directly related to an ENTITY in the project's function group. We use the AFAS Profit Knowledge Base[1] and the training *Financieel Procesbeheer* provided by AFAS Software as a guidance in the processes that are commonly performed by users of the software product.

DECISION POINT:
We do not include the ENTITY *E-factuur* as an ENTITY in the scope of this migration project. We do, however, include the FUNCTIONALITY to transform the *E-facturen* in the system into an instantiation of an INKOOPFACTUUR. Yet, we do not include FUNCTIONALITY on how to get these ENTITIES into the system.

[1] http://kb.afas.nl

During the identification of FUNCTIONALITY, thinking in processes and the consult with the STAKEHOLDER, we identified an interesting property of the ENTITIES, concerning the sequentiality of ENTITIES. By this, we emphasize the position of the ENTITY around the FUNCTIONALITY on the central entity ACTIEF.

For instance, looking at the entity MEDEWERKER, we can assign an ACTIEF to a MEDEWERKER during the creation of a new ACTIEF, as well as assigning an ACTIEF to a MEDEWERKER when the ACTIEF already exists and is being deprecated. However, from a business process perspective, it does not make sense to order new phones (ACTIEF) if you are not yet sure to whom you will assign these phones, and how many you need. Therefore, the entity MEDEWERKER is prior to the process or creating a new ACTIEF.

Also, it is interesting to see which point of entry users of the software product use to perform a certain FUNCTIONALITY. For instance, when we assign an ACTIEF to a MEDEWERKER, the point of entry of the process can be at the ACTIEF, or at the MEDEWERKER, since both entities allow the assignment. The actual point of entry should play a role during the prioritization of MIGRATION MAPPINGS. The determination of the common point of entry is supported by the AFAS Knowledge Base documentation and the STAKEHOLDERS involved in the migration project.

Define entity relationships

We have analyzed the relationships between the entities through means of analysis of the software documentation and interface. The relationships have been visualized in Figure E.5. We have not included detail on the specifications of the relationships, such as multiplicity, direction and labeling, since this does not serve the functional scope of the case study.

In Figure E.7, we present the DATA MODEL as documented in formal design documents of the case company. This DATA MODEL does show details on the relationships. The figure has been extracted from an internal document entitled *RFI00212 Vaste Activa Optimalisatie.docx*, available on the case company's SharePoint intranet. Entities and relationships that are colored in red are new tables and foreign keys which have been added in an optimization project which was described in the document.

Define personas

In this case study, we have used a fictitious organization entitled Clean-X to described personas and their roles within that organization. Given the fact that the context of the software product is very broad, based on the variety of applications in different types of organizations, we believed it would be best to use a fictitious organization which has little to do with software and technology. This is why we have introduced the organization Clean-X, which specializes in the outsourcing of cleaning activities at office buildings.

By using a type of organization which has little affection with technology and software, we aim to prevent bias, based on the definition of personas which are familiar with modern forms of information and communication technology.

This case study knows the following PERSONAS:

EMPLOYEE An EMPLOYEE (Dutch: *Medewerker*) is the most generic form of the users of an internal software platform such as AFAS InSite. A requirement for the persona is that the EMPLOYEE is in an active state of employment at the organization at which the software product is deployed.

Characteristics John Pierson is a 42 years old male and works as a window cleaner at a medium sized organization called Clean-X. The business specializes in the outsourcing of cleaning activities at office buildings. John has already worked for this organization for 20 years, and still enjoys each day he is able to step onto the ladders and leave the area with perfectly clean and transparent windows. John was born and raised in Belgium, and has moved to the Netherlands when he found his passion in the cleaning of windows and got employed at Clean-X. John is now married for 18 years to a Dutch lady called Ellen and together they have one son, Jason, born in 1998.

Needs and goals John's goal is to maintain the steady pace in his life he is currently following, without having to change too many things. He does not have any ambition to promote to a managerial function, neither does he wish to become team leader or educate new employees. Simply put, John is happy with the current work he is doing, and he does not envision a drastic change in work atmosphere. His needs are therefore focused on retaining the current pace and not having to change too many things about the way he performs his work. This implies that all too many technological advancements are hard for John to cope with.

Skills and competencies As John was born in the year 1972, he has not grown up with the affinity with technology as most youngsters have in this era. He only uses a computer for entertainment and emailing, and does not need to use a computer for his daily work routine.

Constraints John does not have enough with software products and can thus not rely sufficiently on his intuition when interacting with user interfaces. He has a long learning curve and struggles with anything outside of his comfort zone.

FACILITY MANAGER A FACILITY MANAGER (Dutch: *facilitair manager*) manages the internal facilities and facility staff of an organization. The manager is responsible for the availability and maintenance of facilities, and can order materials and inventory to make these activities possible.

Characteristics Coert van Haasteren is facility manager at Clean-X. In contrast to most of his colleagues, he is responsible for the facilities inside of their own organization, instead of the cleaning of facilities at other organizations by means of outsourcing. Coert is currently 53 years old has a wife and three children. He has studied Facility management at the Hogeschool van Rotterdam and has worked in facility management for 12 years already.

Needs and goals Coert is looking for ways to make his work easier. Around the millennium, he has come to the understanding that software products can assist both individuals and organizations in both effectiveness and efficiency. As he believes facility management is becoming more technological-centered, his goal is to put this trend to good use. Retirement is not something he is currently concerned about, he loves what he does and wishes to keep doing it for as long as possible.

Skills and competencies Coert is a real team player. By delegating tasks to his facility staff, he gets the most out of his people, and always makes sure everyone is united in the goals they need to achieve as a team. Also, Coert understands the people-aspect to management, and is therefore also concerned about the well-being of his staff. Coert is also eager and able to learn about the technological support in manual labor.

Constraints Coert is constrained by the fact that he can not learn about functionality in a software product as fast as his younger co-workers can. The younger generation seems to be able to rely on intuition when discovering new software products or user interfaces, whereas Coert struggles more with becoming familiar with the interfaces.

ICT MANAGER An ICT MANAGER acts as a systems administrator in an organization. He is responsible for the configuration, maintenance and support for internal information and communication technology. This includes both overall systems, as well as systems that are facilitated for a particular individual, such as a smartphone or computer.

Characteristics Niels Hogendorp is an ICT manager at Clean-X, facilitating ICT for the employees of the organization. Niels is a single 24 year old male, without any children. He has started to work for Clean-X one year ago. Niels lives close to the head office of Clean-X, which makes him feel flexible in his working hours. This results in the fact that Niels can often still be found in the office in the evening, finishing up on projects he is excited about.

Needs and goals In his daily work, Niels' goal is to empower his colleagues by means of technology. However, the ICT landscape of Clean-X has become very diffused over the years. Therefore, Niels has the urge to reduce the diffusion of technological alternatives and envisions a more standardized ICT landscape. To increase the efficiency in the work he does, Niels tries to capture information and knowledge in manuals, such that employees of Clean-X become more able to perform self-service instead of relying on the ICT manager at all time.

Skills and competencies Niels is an analytic person who works best on an individual basis with clear goals to achieve. He does not mind working in teams, yet he needs to know what his responsibilities and tasks are, so that he can work towards those goals on an individual basis. Niels always takes the time to explain himself and his ideas to others, in order to help others to proceed. Because Niels has affinity with technology, he is always eager to learn more and new things. He therefore does not mind to spend his private time on exploring new things, even though it is work related.

Constraints Niels is not a team player, as he prefers to work on himself on his individual tasks. Niels is fine with the work he is currently doing, and does not envision to promote to a higher position in the organization

FINANCIAL CONTROLLER A FINANCIAL CONTROLLER is an expert on finance and economics who supports the management and directors by planning and controlling the organizational processes.

Characteristics Eva Vogels is a 30 years old female employee at Clean-X who has worked as a financial controller for over 10 years. Her dedication to her work has never interfered with her personal situation at home, with her husband and two daughters. Ellen loves working with numbers and diving deeper into details when something does not add up. Ellen does not mind to step up in order to reduce the workload on one of her co-workers. She is a true team player.

Needs and goals Ellen's goals are to provide the financial team of the organization with financial statistics that are correct, reliable and up-to-date, in a timely manner. This allows the organization to make strategic decisions based on her data. She strives for the organizational goal of net profit maximization, while retaining the quality of the work they deliver.

Skills and competencies Ellen is a team player with affinity for numbers and statistics. Nothing is left to coincidence when she is responsible for the work. Ellen has the potential to grow within the organization and stand by the greater minds of the strategic decisions.

Constraints Ellen is constrained by the fact that she sometimes struggles with change to her routines. This causes her to lose control and overview, which makes her feel inconvenient. Therefore, Ellen prefers for things to remain as they are, rather than to change routines.

CHIEF FINANCIAL OFFICER A CHIEF FINANCIAL OFFICER is responsible managing the financial risks of an organization, in order to make the organization as profitable as possible. The individual is responsible for the strategic planning, controlling and reporting of the organization's financial results.

Characteristics Jack van Deur is the chief financial officer (CFO) of the Clean-X organization. Together with the current CEO, he founded the organization 30 years ago. With the age of 63, Jack is also the oldest member of the organization's board. Jack is married, has a son and a daughter, and one grandson. Jack was born in Germany and moved to the Netherlands when he decided to start the Clean-X business with the CEO.

Needs and goals Jack's goal and objective is to keep the organization profitable, and maintain a steady grip on the financial results of the organization. He wishes to keep gaining more and more timely insight in the financial performance of the organization, and is exploring technological solutions to do so. He needs transparency, not only for himself but also for the rest of the organization.

Skills and competencies Jack is a real team player and is capable of motivating colleagues to get their work done on an effective and efficient basis. He is a born leader and also takes a personal interest in the lives of his employees. He relies on his team of controllers to define and shape strategy and translate this into tactics and operations.

Constraints Jack is very limited in scope to his financial figures. He does not take an interest in elaborated situations, as long as the figures turn out positive.

Define software platforms

The AFAS Profit software suite facilitates multiple SOFTWARE PLATFORMS, as follows:

- AFAS Profit Windows
- AFAS InSite
- AFAS OutSite
- AFAS Pocket iOS
- AFAS Pocket Android

We discuss each of these SOFTWARE PLATFORMS and elaborate on their relevance in this case study. Based on the description, we either include or exclude them from this case study. An important qualifier for the inclusion of a SOFTWARE PLATFORM is that one of the identified PERSONAS must use the FUNCTIONALITY from within the scope of this migration project

AFAS Profit Windows This SOFTWARE PLATFORM is the initial platform from which we aim to migrate the functionality to other platforms. It is therefore central in the migration project. Yet, since we do not aim to migrate FUNCTIONALITY to this platform, we do not elaborate on the SOFTWARE PLATFORM in this section.

AFAS InSite The SOFTWARE PLATFORM *AFAS InSite* is the web application of the AFAS Profit software suite which enables self service for employees and managers of an organization. By authenticating as an employee of the organization, the user gets access to his personal details in the AFAS Profit database, and any other shared and authenticated data that is stored about the organization's administration.

Platform type In Section 4.5.2, we describe how we apply and extend the software ecosystem taxonomy by Bosch (2009) in order to design a classification of software platforms. Based on this classification, we can identify the AFAS InSite software platform as being a **web** platform. This implies that from a user's perspective, the software product can run from a modern web browser, without the necessity of installing specific components that are not default in modern browsers.

A platform typology by Gawer (2009) is also described in Section 4.5.2. This typology classifies the AFAS InSite software platform as being an **internal platform**, because it is merely used for employees within the organization with an active state of employment. No external parties are involved in the AFAS InSite software platform.

SWOT analysis

– **Strengths**: A great advantage and thus strength of this software platform is the fact that the software application can run in any modern web browser. Web browsers are present in a multitude of devices, such as computers, laptops, tablets and smartphones, which introduces a widespread support for the platform. Also, the platform does not require the installation of any third party plugins which are not default in all modern web browsers, such as Adobe Flash, Java or Microsoft Silverlight. Because, just as most web-applications, AFAS InSite is based on a client-server software architecture, a large portion of the computing is performed on the server-side. This reduces the workload on the client in the web browser. Since the composition and functionality of the client's web interface is provided by the server on every request, updates are automatically provided to the client, without the necessity of updating the client application.

– **Weaknesses**: Because the client-server software architecture is highly dependent on a stable connection between the clients and server, an absent or instable internet connection can be fatal to a correct functioning of the application. The same goes for the fact that the client-server architecture is highly dependent on the available resources at the server-node. If the workload on the server becomes too high, all clients are affected, yet none can support the server by taking over their portion of the workload.

– **Opportunities**: As the AFAS InSite software platform is supported by any modern web browser, new devices and types of devices with a web browser are often supported by default. For instance, an e-reader with web browsing functionality also supports AFAS InSite, as well as a Samsung Smart TV and Sony PlayStation.

– **Threats**: The AFAS InSite client depends on the assumption that any modern web browser supports web standards. If a browser or device does not correctly adhere to the standards, the web interface could be erroneous. Supporting a multitude of different interpretations of "web standards" challenges the AFAS InSite designers in providing the same web interface for any device and browser.

Functional context and constraints The functional context of the AFAS InSite software platform focuses mainly on the portability of the application that runs in the web browser. Since any modern web browser can support the web application, many devices support the software by default. This does not only include computers, laptops and tablets, but also smartphones, e-readers, smart TVs and gaming consoles. Since the software product does not require the installation of additional software, the web application can be used at any time, at any place, as long as the device has a web browser and stable internet connection.

Also, the AFAS InSite software platform required authentication for any user. This implies that the functionality within the software product can assume an interaction with a trusted and authenticated party at all time. Because the access rights are determined by authentication, functionality can be made available to specific users and user groups, based on their role within the organization.

The software platform's functionality is constrained by the fact that authentication is required to visit any page of the system. This means that only the login and "forgot password" page are available, but not even a "page not found" or "frequently asked questions" page is available. For non-authenticated users, such as users with no active state of employment, this means they have no access to their personal data, nor can they read up on additional information about a possible reason why they have no access to the system.

Technical context and constraints The technical context of the software platform constrains the interfacing of the web application with peripheral devices of the device that runs the web browser. Web browsers are by default protected against interfering with any software outside the browser's context. This is an obvious security precaution, yet it limits the attachment of peripheral devices such as handheld barcode scanners, document scanners and webcams microphones. Luckily, such features and interfaces become more popular, and new technology standards such as HTML5 support interfacing with such devices.

As described in the SWOT analysis of the software platform, the technical context allows updates to be automatically spread amongst clients, as the server determines the contents and functionality of the web interface.

Also, as previously discussed on the SWOT analysis of the software platform, the web application is highly dependent on the availability and performance of the application server. No computing of the server can be distributed amongst clients.

AFAS OutSite Of the identified PERSONAS, all individuals are in an active state of employment for the organization. This means they are able to work with internal systems of the organization. AFAS OutSite is a portal application for individuals outside of the organization. This means that the SOFTWARE PLATFORM does not suit the types of PERSONAS of this migration project.

The FUNCTIONALITY in the function group of this case study focuses around *Vaste activa*. This ENTITY is purely an internal object of the organization. It is manager by the organization itself, no users from outside the organization have access to the FUNCTIONALITY of this ENTITY. Therefore, the SOFTWARE PLATFORM AFAS OutSite does not qualify for FUNCTIONALITY of *Vaste activa*.

The procurement and divestment of *Vaste activa* does involve external parties, although these do not directly act on the ENTITIES within the facilitating organization. The *Vaste activa* are delivered by and to external parties, but the maintenance on the ENTITIES in the software is done by employees.

AFAS Pocket for iOS The AFAS Pocket smartphone application originates from the Employee Self Service (ESS) philosophy of AFAS Software, where the recording of data is done at its source, instead of through intermediaries. The application allows an employee to enter worked hours into the system, to claim declarations for work-related activities, search through the CRM database, request furlough and submit sick leave.

However, the SOFTWARE PLATFORM does not suit the processes that surround the FUNCTIONALITY in the function group *Vaste activa*. The application is designed for common FUNCTIONALITY that needs to be performed quickly without the need of having to grab a computer with a browser and going through authentication. The processes an FUNCTIONALITY of *Vaste activa* does not qualify. Therefore, this SOFTWARE PLATFORM does not suit this case study.

AFAS Pocket for Android As with the SOFTWARE PLATFORM *AFAS Pocket iOS*, the *AFAS Pocket Android* smartphone application does not suit the FUNCTIONALITY of *Vaste activa*. Therefore, this SOFTWARE PLATFORM is also excluded from this case study.

Combine personas with software platforms

In the previous sections, we have instantiated the template concepts PERSONA and SOFTWARE PLATFORM. The FUNCTIONALITY in the function group of this case study is focused around an ENTITY which is primarily handled inside an organization's administration, without involvement of external parties. Therefore, only the SOFTWARE PLATFORM AFAS InSite qualifies for this case study. The SOFTWARE PLATFORM was designed to empower every employee with self service. This implies that any of the PERSONAS we have identified are combined with this SOFTWARE PLATFORM to create a SCENARIO.

Identify mapping candidates

To provide a user-friendly interface to interact with the FUNCTIONALITY and identify MIGRATION MAPPING candidates, we designed a user interface which displays a table with all required data. As each row represented a set of functionality, the columns represented the ENTITY, function group, CRUD-operator, FUNCTIONALITY, and different SCENARIOS. The cells under the columns of the SCENARIOS represent the MIGRATION MAPPINGS and candidates.

For each row, and thus for each FUNCTIONALITY, we assessed whether the FUNCTIONALITY could serve as a potential MIGRATION MAPPING in the given SCENARIO. This was done in a closed environment on an individual basis, as consulting other STAKEHOLDERS would be performed in the upcoming activities *Review mapping candidates* and *Prioritize mapping candidates*.

The creation of the concept MIGRATION MAPPING in this activity means that the FUNCTIONALITY serves a mapping candidate for the given SCENARIO. No *Priority* was yet assigned to the instantiated concept. When relevant, we have also attached a DESIGN RATIONALE to the MIGRATION MAPPING, which captures the underlying decision made about the MIGRATION MAPPING. Figure E.2 displays the user interface of the tool we have designed for this activity and case study.

Gegeven	Functiegroep	CRUD	Functionaliteit	Employee AFAS InSite	Procurement agent AFAS InSite	Facility manager AFAS InSite	ICT manager AFAS InSite	Financial controller AFAS InSite	Chief financial officer AFAS InSite
Actief	Vaste activa	C	Actief toevoegen						
Actief	Vaste activa	C	Actief toevoegen via Inkoopfactuur						
Actief	Vaste activa	C	Nieuwe Vaste activa importeren						
Actief	Vaste activa	R	Actief weergeven						
Actief	Vaste activa	R	Activa van een Activagroep raadplegen						
Actief	Vaste activa	R	Activa van een Medewerker raadplegen						
Actief	Vaste activa	R	Controleoverzicht vaste activa boekhouding raadplegen						
Actief	Vaste activa	U	Actief bewerken						
Actief	Vaste activa	U	Wijzigen activacode						
Actief	Vaste activa	U	Herberekenen cumulatieven vaste activa						
Actief	Vaste activa	U	Gekoppeld actief aan actief toevoegen						
Actief	Vaste activa	U	Activa koppelen aan een verzekering						
Actief	Vaste activa	D	Actief verwijderen						
Activa stamkaart	Vaste activa	R	Activa stamkaart raadplegen						
Activagroep	Vaste activa	C	Activagroep toevoegen						
Activagroep	Vaste activa	C	Nieuwe Activagroepen importeren						
Activagroep	Vaste activa	R	Activagroep weergeven						
Activagroep	Vaste activa	U	Activagroep bewerken						
Activagroep	Vaste activa	U	Wijzigen activagroepcode						
Activagroep	Vaste activa	D	Activagroep verwijderen						
Afbeelding	Algemeen	CU	Afbeelding plakken						
Afbeelding	Algemeen	C	Afbeeldingen importeren						
Afbeelding	Algemeen	C	Afbeelding toevoegen						
Afbeelding	Algemeen	R	Afbeelding weergeven						

Figure E.2: MIGRATION MAPPING tool

In Figure E.5, we present an example of the MIGRATION MAPPING with preliminary candidates.

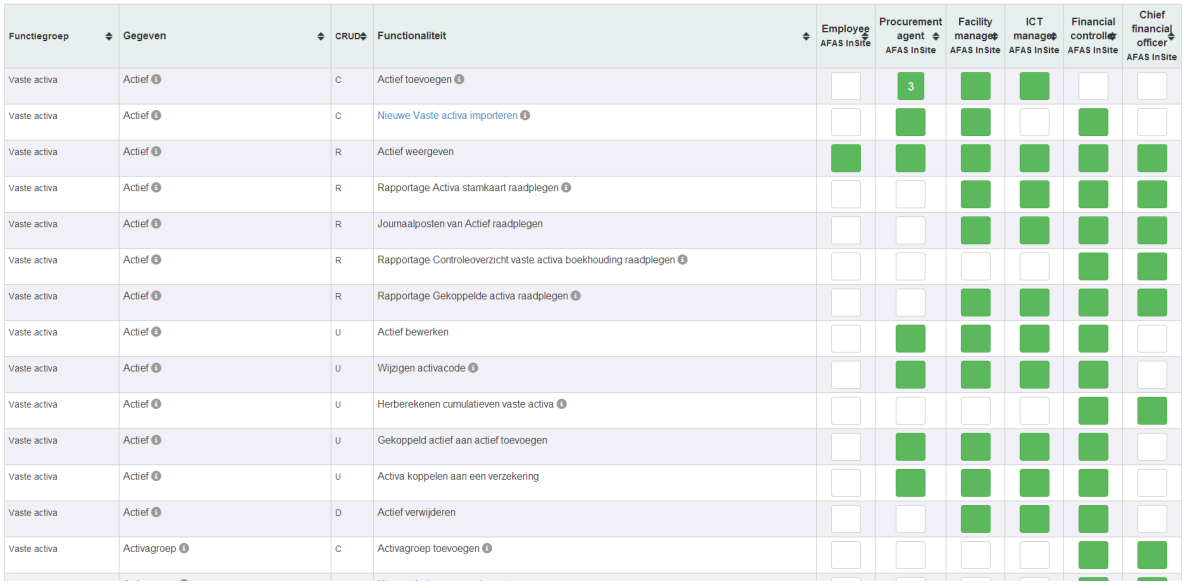


Figure E.3: MIGRATION MAPPING tool with candidates

Review mapping candidates—Prioritize mapping candidates

In a two hour group session with the stakeholders MOHAMED AMRI and HENK VAN DER SCHUUR, both a product manager at AFAS Software, we reviewed and prioritized the MIGRATION MAPPING candidates that were selected in the previous activity. The same tool was used to present the candidates and give an overview of the progress and results of the group session. The entire session was recorded by audio, so that the discussions of the group session could be consulted in a later stage.

The input of the group session was a total of 105 sets of FUNCTIONALITY, originating from 33 ENTITIES. During discussions in the group session, 5 ENTITIES were excluded, which implied the exclusion of 26 sets of FUNCTIONALITY. These ENTITIES were excluded because they did not provide significant relevance to the identified SCENARIOS. Of the remaining 79 sets of FUNCTIONALITY, originating from 28 ENTITIES, 56 sets of FUNCTIONALITY were assigned a priority by means of 131 MIGRATION MAPPINGS. This means that 23 sets of FUNCTIONALITY have no MIGRATION MAPPING assigned, and are therefore also excluded from the migration.

These 56 sets of FUNCTIONALITY with a total of 131 MIGRATION MAPPINGS and their priorities are included as input for the VISUALIZATIONS of the method's output.

Select stakeholders

The outcome of the group session in which we assigned priorities to the MIGRATION MAPPING instantiations, will be presented to two product managers of AFAS Software. They have also been identified as stakeholders in Section E.2.2, named MOHAMED AMRI and HENK VAN DER SCHUUR. For now, these two stakeholders are the only stakeholders for whom we design a visualization of this case study. In a later stage, for instance during the actual composition of the product roadmap, another visualization may need to be designed.

Design visualization

As with the previous template method instantiation, a matrix of functionality, scenarios and mapping priorities suits the goal of this case study best. The stakeholders request a prioritized list of requirements, which can act as input for their product roadmap.

Each row of the visualization contains a set of functionality, with its corresponding functionality classifier, entity and function group. The other columns include the five scenarios of the case study, the sum of the mapping priorities, and the average priority, based on a division of the sum by the total number of scenarios.

The spreadsheet has been sorted on the final column at first, which contains the average priority of the functionality. Second sorting is done by the corresponding entity and CRUD-operator.

To support interpretation by visual aids in the spreadsheet, the variables concerning the mapping priority are given a background color, depicting its value in correspondence with other values in the column. The more green a cell is, the higher its value in relation with the other cells in the column.

E.2.2 Concept instantiations

MIGRATION PROJECT DEFINITION

MIGRATION PROJECT DEFINITION	
Code	AFAS.Profit.ACT.001
Scope	*Vaste activa*
Start date	7 April 2014
Version	1.0
Goal	Migration of function group Vaste activa to AFAS InSite

Table E.3: MIGRATION PROJECT DEFINITION

STAKEHOLDER

GERARD NIJBOER	
Role	Participant

Table E.4: STAKEHOLDER

HENK VAN DER SCHUUR	
Role	Informer

Table E.5: STAKEHOLDER

MOHAMED AMRI	
Role	Observer—Informer

Table E.6: STAKEHOLDER

JAN GRIJZEN	
Role	Informer

Table E.7: STAKEHOLDER

DOMAIN ONTOLOGY

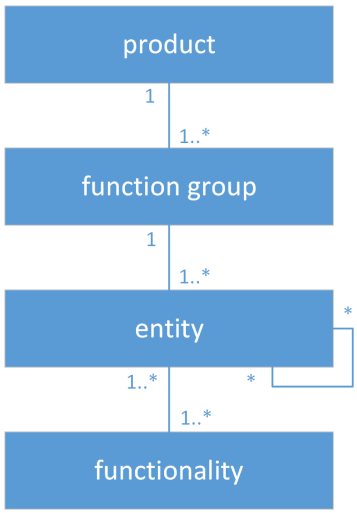


Figure E.4: DOMAIN ONTOLOGY

ENTITY

Product	Function group	ENTITY
Vaste activa	Vaste activa	AFSCHRIJVINGSMETHODE
Vaste activa	Vaste activa	ACTIEF
Vaste activa	Vaste activa	ACTIVA STAMKAART
Vaste activa	Vaste activa	ACTIVAGROEP
Algemeen	Administratie	ADMINISTRATIE
Algemeen	Algemeen	AFBEELDING
Algemeen	Algemeen	BIJLAGE
Vaste activa	Vaste activa	COMMERCIËLE AANSCHAFFINGSSTAAT
Vaste activa	Vaste activa	COMMERCIËLE AFSCHRIJVINGSSTAAT
Vaste activa	Vaste activa	COMMERCIËLE AFSCHRIJVINGSTERMIJN

Product	Function group	ENTITY
Financieel	Crediteur	CREDITEUR
Financieel	Financieel	DAGBOEK
CRM	Dossier	DOSSIERITEM
Vaste activa	Vaste activa	FACTUUR
Financieel	Financiële mutaties	FINANCIËLE MUTATIE
Vaste activa	Vaste activa	FISCALE AANSCHAFFINGSSTAAT
Vaste activa	Vaste activa	FISCALE AFSCHRIJVINGSSTAAT
Vaste activa	Vaste activa	FISCALE AFSCHRIJVINGSTERMIJN
Financieel	Grootboek	GROOTBOEKREKENING
Logistiek	Facturen	INKOOPFACTUUR
Vaste activa	Vaste activa	INVESTERING
Payroll	Payroll definitief	JOURNAALPOST
CRM	Organisatie/persoon	MEDEWERKER
CRM	Organisatie/persoon	ORGANISATIE/PERSOON
Algemeen	Inrichting	PERIODETABEL
Algemeen	Inrichting	PERIODEVERDEELTABEL
Vaste activa	Vaste activa	SUBSIDIE
Vaste activa	Vaste activa	VERKOOP
Vaste activa	Vaste activa	VERZEKERING
Algemeen	Inrichting	VRIJE TABEL
Vaste activa	Vaste activa	WIJZIGING ACTIEF

Table E.8: ENTITY

FUNCTIONALITY

ENTITY	CRUD	FUNCTIONALITY
ACTIEF	C	Actief toevoegen
ACTIEF	C	Nieuwe Vaste activa importeren
ACTIEF	R	Actief weergeven
ACTIEF	R	Rapportage Activa stamkaart raadplegen
ACTIEF	R	Journaalposten van Actief raadplegen
ACTIEF	R	Rapportage Controleoverzicht vaste activa boekhouding raadplegen
ACTIEF	R	Rapportage Gekoppelde activa raadplegen
ACTIEF	U	Actief bewerken
ACTIEF	U	Wijzigen activacode
ACTIEF	U	Herberekenen cumulatieven vaste activa
ACTIEF	U	Gekoppeld actief aan actief toevoegen
ACTIEF	U	Activa koppelen aan een verzekering
ACTIEF	D	Actief verwijderen
ACTIVAGROEP	C	Activagroep toevoegen

ENTITY	CRUD	FUNCTIONALITY
ACTIVAGROEP	C	Nieuwe Activagroepen importeren
ACTIVAGROEP	R	Activagroep weergeven
ACTIVAGROEP	R	Activa van een Activagroep raadplegen
ACTIVAGROEP	U	Activagroep bewerken
ACTIVAGROEP	U	Wijzigen activagroepcode
ACTIVAGROEP	D	Activagroep verwijderen
AFBEELDING	CD	Afbeelding toevoegen
AFBEELDING	CRUD	Afbeelding kopiëren naar ander doel
AFBEELDING	C	Afbeelding importeren
AFBEELDING	R	Afbeelding weergeven
AFBEELDING	R	Afbeelding tonen in groot formaat
AFBEELDING	U	Toelichting van afbeelding bewerken
AFBEELDING	D	Afbeelding verwijderen
BIJLAGE	C	Bijlage toevoegen aan inkoopfactuur
COMMERCIËLE AANSCHAFFINGSSTAAT	R	(Commerciële) Aanschaffingsstaat raadplegen
COMMERCIËLE AFSCHRIJVINGSSTAAT	R	(Commerciële) Afschrijvingsstaat raadplegen
COMMERCIËLE AFSCHRIJVINGSTERMIJN	R	(Commerciële) Afschrijvingstermijn raadplegen
CREDITEUR	R	Crediteur weergeven
DAGBOEK	R	Dagboek weergeven en openen
DOSSIERITEM	C	Dossieritem toevoegen
DOSSIERITEM	C	Subdossier toevoegen aan dossieritem
DOSSIERITEM	C	Bijlage aan dossieritem toevoegen
DOSSIERITEM	R	Dossieritem weergeven
DOSSIERITEM	U	Dossieritem bewerken
DOSSIERITEM	U	Bestemming van dossieritem bewerken
DOSSIERITEM	D	Dossieritem verwijderen
DOSSIERITEM	D	Subdossier van dossieritem verwijderen
FACTUUR	C	Factuur aan Actief toevoegen
FACTUUR	R	Factuur van Actief weergeven
FACTUUR	U	Factuur van Actief bewerken
FACTUUR	D	Factuur van Actief verwijderen
FISCALE AANSCHAFFINGSSTAAT	R	Fiscale aanschaffingsstaat raadplegen
FISCALE AFSCHRIJVINGSSTAAT	R	Fiscale afschrijvingsstaat raadplegen
FISCALE AFSCHRIJVINGSTERMIJN	R	Fiscale afschrijvingstermijn
FISCALE AFSCHRIJVINGSTERMIJN	U	Fiscale afschrijvingstermijn bewerken
GROOTBOEKREKENING	C	Grootboekrekening toevoegen
GROOTBOEKREKENING	R	Grootboekrekening weergeven
GROOTBOEKREKENING	U	Grootboekrekening bewerken
GROOTBOEKREKENING	D	Grootboekrekening verwijderen
INKOOPFACTUUR	C	Inkoopfactuur toevoegen via inkoopboeking

ENTITY	CRUD	FUNCTIONALITY
INKOOPFACTUUR	C	Inkoopfactuur boeken vanuit ingelezen e-factuur
INVESTERING	C	Investering aan Actief toevoegen
INVESTERING	R	Investering van Actief weergeven
INVESTERING	U	Investering van Actief bewerken
INVESTERING	D	Investering van Actief verwijderen
JOURNAALPOST	UD	Journalisering van (geselecteerde) Activa terugdraaien
JOURNAALPOST	C	Journaalposten opnieuw genereren
JOURNAALPOST	C	Vaste activa journaliseren
JOURNAALPOST	C	Journaliseren afschrijvingen via kolommenbalans van Verslagleggingscockpit
MEDEWERKER	C	Medewerker toevoegen
MEDEWERKER	R	Activa van een Medewerker raadplegen
MEDEWERKER	R	Medewerker weergeven
MEDEWERKER	R	Stamkaart medewerker weergeven
PERIODETABEL	C	Periodetabel toevoegen
PERIODETABEL	C	Jaren toevoegen aan Periodetabel
PERIODETABEL	R	Periodetabel weergeven
PERIODETABEL	U	Periodetabel bewerken
PERIODETABEL	U	Jaren blokkeren in Periodetabel
PERIODETABEL	U	Jaren deblokkeren in Periodetabel
PERIODETABEL	U	Opnieuw opbouwen van Periodetabel
PERIODETABEL	D	Periodetabel verwijderen
PERIODETABEL	D	Jaren verwijderen van Periodetabel
PERIODEVERDEELTABEL	C	Periodeverdeeltabel toevoegen
PERIODEVERDEELTABEL	R	Periodeverdeeltabel weergeven
PERIODEVERDEELTABEL	U	Periodeverdeeltabel bewerken
PERIODEVERDEELTABEL	D	Periodeverdeeltabel verwijderen
SUBSIDIE	C	Subsidie aan Actief toevoegen
SUBSIDIE	R	Subsidie van Actief weergeven
SUBSIDIE	U	Subsidie van Actief bewerken
SUBSIDIE	D	Subsidie van Actief verwijderen
VERKOOP	C	Actief verkopen
VERKOOP	R	Actief verkoop weergeven
VERKOOP	U	Actief verkoop bewerken
VERKOOP	D	Actief verkoop verwijderen
VERZEKERING	C	Nieuwe Vaste activa Verzekeringen importeren
VERZEKERING	C	Verzekering toevoegen
VERZEKERING	R	Verzekering weergeven
VERZEKERING	U	Verzekering bewerken
VERZEKERING	D	Verzekering verwijderen
VRIJE TABEL	C	Type subsidie toevoegen

ENTITY	CRUD	FUNCTIONALITY
VRIJE TABEL	C	Locatie actief toevoegen
VRIJE TABEL	R	Type subsidie weergeven
VRIJE TABEL	R	Locatie actief weergeven
VRIJE TABEL	U	Type subsidie bewerken
VRIJE TABEL	U	Locatie actief bewerken
VRIJE TABEL	D	Type subsidie verwijderen
VRIJE TABEL	D	Locatie actief verwijderen
WIJZIGING ACTIEF	R	Wijzigingen van een Actief weergeven

Table E.9: FUNCTIONALITY of ENTITIES in Vaste activa

DATA MODEL

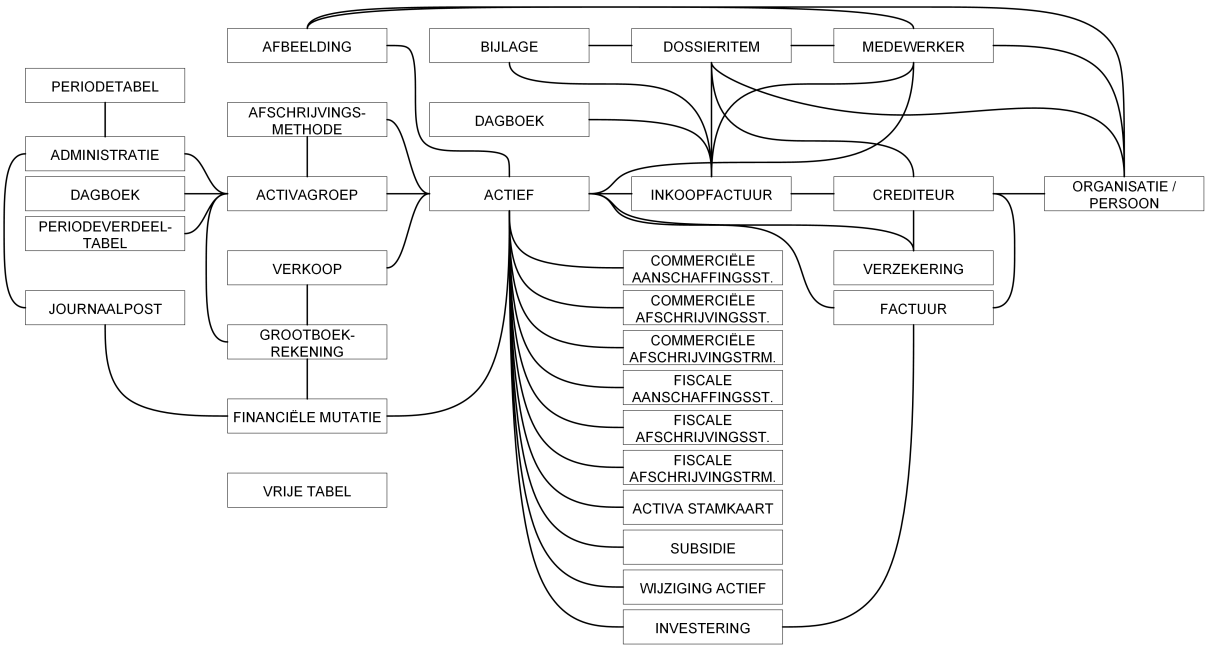


Figure E.5: DATA MODEL

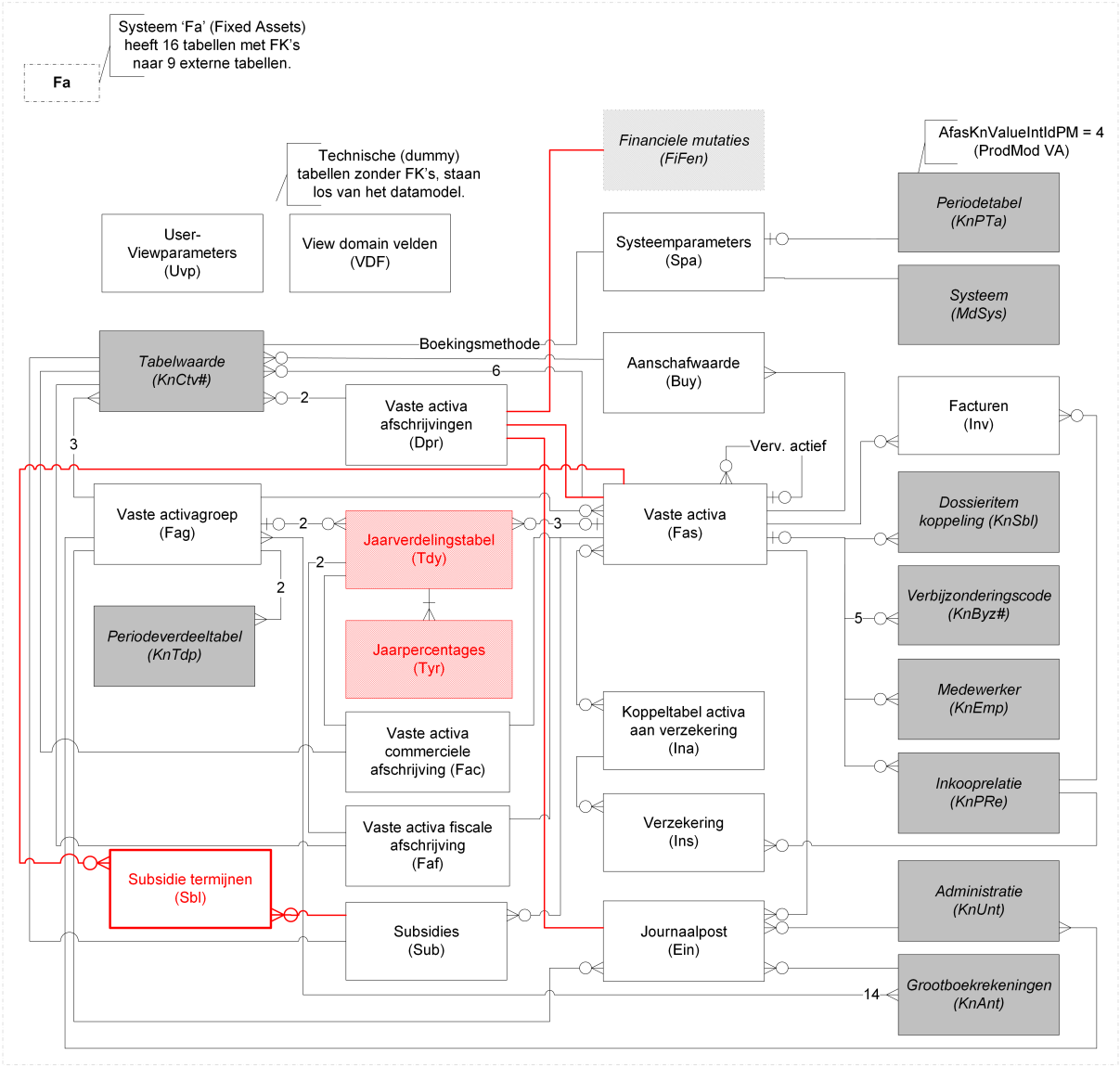


Figure E.6: DATA MODEL by formal design document

PERSONA

JOHN PIERSON	
Role	Employee

Table E.10: PERSONA

COERT VAN HAASTEREN	
Role	Facility manager

Table E.11: PERSONA

NIELS HOGENDORP	
Role	ICT manager

Table E.12: PERSONA

EVA VOGELS	
Role	Financial controller

Table E.13: PERSONA

JACK VAN DEUR	
Role	Chief financial officer

Table E.14: PERSONA

SOFTWARE PLATFORM

AFAS INSITE	
Platform type (Bosch, 2009)	Web
Platform type (Gawer, 2009)	Internal platform

Table E.15: SOFTWARE PLATFORM

EMPLOYEE on AFAS INSITE	
Persona	EMPLOYEE
Software platform	AFAS INSITE

Table E.16: SCENARIO

PROCUREMENT AGENT on AFAS INSITE	
Persona	PROCUREMENT AGENT
Software platform	AFAS INSITE

Table E.17: SCENARIO

FACILITY MANAGER on AFAS INSITE	
Persona	FACILITY MANAGER
Software platform	AFAS INSITE

Table E.18: SCENARIO

ICT MANAGER on AFAS INSITE	
Persona	ICT MANAGER
Software platform	AFAS INSITE

Table E.19: SCENARIO

FINANCIAL CONTROLLER on AFAS INSITE	
Persona	FINANCIAL CONTROLLER
Software platform	AFAS INSITE

Table E.20: SCENARIO

CHIEF FINANCIAL OFFICER on AFAS INSITE	
Persona	CHIEF FINANCIAL OFFICER
Software platform	AFAS INSITE

Table E.21: SCENARIO

VISUALIZATION

Functiegroep	Gegeven	CRUD	Functionaliteit	Employee	Facility manager	ICT manager	Financial controller	Chief financial officer	SUM	COUNT	AVG	AVG/5
Algemeen	Afbeelding	R	Afbeelding weergeven	5	5	5	5	5	25	5	5,0	5,0
Vaste activa	Actief	R	Actief weergeven	4	4	4	5	5	22	5	4,4	4,4
Vaste activa	Actief	C	Actief toevoegen		5	5	5		15	3	5,0	3,0
Algemeen	Afbeelding	CD	Afbeelding toevoegen		5	5	5		15	3	5,0	3,0
Algemeen	Afbeelding	D	Afbeelding verwijderen		5	5	5		15	3	5,0	3,0
Organisatie/persoon	Medewerker	R	Activa van een Medewerker raadplegen	5			5	5	15	3	5,0	3,0
Vaste activa	Actief	U	Gekoppeld actief aan actief toevoegen		4	4	5		13	3	4,3	2,6
Vaste activa	Actief	U	Actief bewerken Algemeen		2	5	5		12	3	4,0	2,4
Vaste activa	Factuur	R	Factuur van Actief weergeven		3	3	3	3	12	4	3,0	2,4
Dossier	Dossieritem	C	Dossieritem toevoegen	2	2	2	2	2	10	5	2,0	2,0
Dossier	Dossieritem	C	Bijlage aan dossieritem toevoegen	2	2	2	2	2	10	5	2,0	2,0
Dossier	Dossieritem	R	Dossieritem weergeven	2	2	2	2	2	10	5	2,0	2,0
Dossier	Dossieritem	U	Dossieritem bewerken	2	2	2	2	2	10	5	2,0	2,0
Crediteur	Crediteur	R	Crediteur weergeven				4	4	8	2	4,0	1,6
Vaste activa	Actief	R	Rapportage Activa stamkaart raadplegen				4	2	6	2	3,0	1,2
Vaste activa	Actief	R	Journaalposten van Actief raadplegen				4	2	6	2	3,0	1,2
Vaste activa	Commerciële aanschaffingsstaat	R	(Commerciële) Aanschaffingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Commerciële afschrijvingsstaat	R	(Commerciële) Afschrijvingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Commerciële afschrijvingstermijn	R	(Commerciële) Afschrijvingstermijn raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Fiscale aanschaffingsstaat	R	Fiscale aanschaffingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Fiscale afschrijvingsstaat	R	Fiscale afschrijvingsstaat raadplegen				3	3	6	2	3,0	1,2
Vaste activa	Fiscale afschrijvingstermijn	R	Fiscale afschrijvingstermijn				3	3	6	2	3,0	1,2
Vaste activa	Investering	R	Investering van Actief weergeven				3	3	6	2	3,0	1,2
Vaste activa	Actief	U	Actief bewerken Financieel				5		5	1	5,0	1,0
Algemeen	Bijlage	C	Bijlage toevoegen aan inkoopfactuur				5		5	1	5,0	1,0
Dossier	Dossieritem	D	Dossieritem verwijderen	1	1	1	1	1	5	5	1,0	1,0
Vaste activa	Actief	D	Actief verwijderen				4		4	1	4,0	0,8
Vaste activa	Factuur	C	Factuur aan Actief toevoegen				4		4	1	4,0	0,8
Journaalpost	Journaalpost	UD	Journalisering van (geselecteerde) Activa terugdraaien				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	C	Subsidie aan Actief toevoegen				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	R	Subsidie van Actief weergeven				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	U	Subsidie van Actief bewerken				2	2	4	2	2,0	0,8
Vaste activa	Subsidie	D	Subsidie van Actief verwijderen				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	C	Actief verkopen				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	R	Actief verkoop weergeven				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	U	Actief verkoop bewerken				2	2	4	2	2,0	0,8
Vaste activa	Verkoop	D	Actief verkoop verwijderen				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	C	Verzekering toevoegen				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	R	Verzekering weergeven				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	U	Verzekering bewerken				2	2	4	2	2,0	0,8
Vaste activa	Verzekering	D	Verzekering verwijderen				2	2	4	2	2,0	0,8
Vaste activa	Investering	C	Investering aan Actief toevoegen				3		3	1	3,0	0,6
Vaste activa	Investering	U	Investering van Actief bewerken				3		3	1	3,0	0,6
Vaste activa	Investering	D	Investering van Actief verwijderen				3		3	1	3,0	0,6
Vaste activa	Actief	U	Verzekering koppelen aan actief				2		2	1	2,0	0,4
Vaste activa	Factuur	U	Factuur van Actief bewerken				2		2	1	2,0	0,4
Vaste activa	Actief	R	Rapportage Controleoverzicht vaste activa boekhouding raadplegen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	C	Type subsidie toevoegen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	C	Locatie actief toevoegen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	R	Type subsidie weergeven				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	R	Locatie actief weergeven				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	U	Type subsidie bewerken				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	U	Locatie actief bewerken				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	D	Type subsidie verwijderen				1	1	2	2	1,0	0,4
Inrichting	Vrije tabel	D	Locatie actief verwijderen				1	1	2	2	1,0	0,4
Vaste activa	Factuur	D	Factuur van Actief verwijderen				1		1	1	1,0	0,2

Figure E.7: VISUALIZATION