WEM: A Design Method for CMS-based Web Implementations

Inge van de Weerd

institute of information and computing sciences, utrecht university
technical report UU-CS-2005-043
www.cs.uu.nl

WEM: A Design Method for CMS-based Web Implementations

Inge van de Weerd
Department of Information and Computing Sciences
Utrecht University, P.O. Box 80089
3508 TB Utrecht, The Netherlands
i.vandeweerd@cs.uu.nl

Abstract

The development of complex, data-intensive web applications is becoming simpler due to the usage of content management systems. Conventional information systems development methods as well as web application development methods do not cover the specific needs of a method for web content management implementations. In this research an assembly-based situational method engineering approach is applied to develop a new design method, called the WebEngineering Method (WEM).

First, a literature research is done to existing design methods and method engineering. Then, the development processes and implementation situations are identified. Consequently candidate methods are selected and analyzed. Finally, a new method of useful method fragments is assembled. By using route map configuration, the method fragments are tailored to obtain situationality.

A meta-modeling technique is proposed that integrates UML activity diagrams and class diagrams for the purpose of analyzing existing methods and assembling the new method. The method developed was validated in a case study, which consisted of a technology testing web application at a large telecommunication organization in the Netherlands. The case study results were promising.

Table of contents

List of figures

List of tables

1 Introduction

A large number of information system development methods are available. Next to established methods like entity-relationship modeling (Chen, 1976) and the more recent Unified Process (Jacobson, Booch & Rumbaugh, 1999), several methods for developing *web* applications have been developed. Examples are WebML (Ceri, Fraternali & Bongio, 2000), UWE (Koch, 2001) and W2000 (Baresi, Garzotto & Paolini, 2001). These methods show significant influences from information systems development methods. This is not surprising since web applications can be seen as a subtype of information systems (Souer, et al. 2005). Gnaho (2001) acknowledges this in his definition of Web applications: a Web application is an Information System providing facilities to access complex data and interactive services via the Web which changes the state of business.

Using data-intensive Web applications raised new problems concerning consistency, navigation, data duplication, content audit and control, tracking of content and mapping the website workflows on the business processes (Vidgen, Goodwin and Barnes, 2001). The solution to these problems was found in content management. A content management system (CMS) makes it possible to create, archive, search, control and publish information from within a flexible and integrated environment (Burzagli, et al., 2004). A special type of content management systems are CMS-based Web applications, which are defined as Web applications for the management and control of information (Souer, et al., 2005).

Developing CMS-based Web applications is a complex process. In academic and professional literature no specific methods on this subject exist. Currently, established information system and Web application development methods are being used, but these methods are not able to cover specific content management aspects.

1.1. Problem definition

The research described in this work is carried out at GX creative online development, a web technology company in the Netherlands. GX is active in the fields of content management, online application development, and integration of backend systems in portal solutions. The company has developed a content management system, GX WebManager, which enables people without a specific technological background in creating, maintaining and integrating several dynamic websites and portals. In addition to their product, the company also develops the web applications that run on GX WebManager.

The implementation of GX WebManager is currently carried out by proprietary methods. However, the need exists to optimize these methods in order to save time and money. Also, the need for a standardized web application development method exists, which can be used by partners of the company. This brings me to my research question:

> *"How should a design method be constructed for the process of developing web applications for GX WebManager?"*

1.2. Relevance

Several well-established design methods for developing information systems exist. In the field of web applications, the research is less mature. Oftentimes, methods are borrowed from the information systems and hypermedia field. However, several methods have been developed over the last 10 years. Examples are WebML, W2000, and several UML extensions. In my research project I will construct an overview of existing methods, as well as develop new methods contrived from earlier ones, specifically for web applications. The scientific relevance can be found in: a) producing a thorough overview on existing standard design methods and their relations, b) the creation of new design methods through method engineering, c)

GX benefits from this research project in the sense that I will provide an inventory of methods that are currently in use at GX; an inventory of existing project situations at GX and a set of usable design methods for WebManager implementations. Furthermore, I will make recommendations for a product extension in order to implement the constructed design methods in WebManager.

1.3. Research methodology

1.3.1. Introduction

This research project is carried out following the design research methodology for performing research in information systems. Design research involves the analysis of the use and performance of designed artifacts to understand, explain and very frequently to improve on the behavior of aspects of Information Systems. (Vaishnavi & Kuechler, 2004). In Figure 1-1 the design cycle followed in this research project is illustrated.

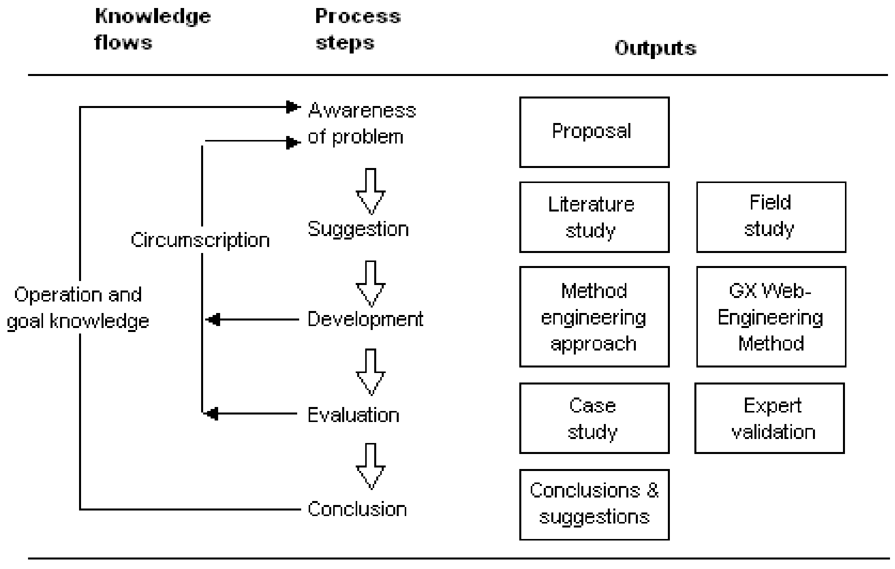


Figure 1-1: Reasoning in the Design Cycle

1.3.2. Knowledge flows

Reasoning in the design cycle is an iterative process, indicated by the knowledge flows, namely *circumscription* and *operation and goal knowledge*. The circumscription process illustrates the knowledge derived from the construction of a certain artifact. When in the development or evaluation phase appears that the proposed suggestion does not work, the researcher is forced back to the first process step. The circumscription flow results in valuable constraint knowledge with which the process can start all over again. The second way of knowledge production in the design cycle is through operation and goal knowledge. Dasgupta and Purao define an operational principle: “any technique or frame of reference about a class of artifacts or its characteristics that facilitates creation, manipulation and modification of artifactual forms”, as cited by Vaishnavi and Kuechler (2004). The conclusion of a research project produces new operation and goal knowledge, which can lead to the awareness of a new problem.

1.3.3. Process steps

Awareness of Problem
At the right side of the diagram the process steps and their outputs are illustrated. First of all, problem awareness is raised from scientific side in the sense that the need existed to conduct research to method engineering in the field of web applications. On the other side, GX wanted to improve its development method they currently use.

Suggestion
The second step is suggestion. During this step a field study and a literature study are carried out. Both studies provide suggestions on how to design methods for WebManager implementations. The literature study embodies a thorough overview of existing methods for developing web applications and information systems. Examples of such methods are WebML (Ceri, Fraternali and Bongio, 2000) and the Unified Software Development Process (Jacobson, Booch & Rumbaugh, 1999). The most suitable methods for this research project are selected and explored in depth.

During the field study an overview is provided of which design method(s) GX is currently using. Furthermore, a research is conducted to find out which types of projects exist within GX. This field research is done in the form of semi-structured interviews, observations and artifact analysis.

The processes at GX and the selected design methods from the literature study are analyzed and saved as method fragments in the method base. The method base is the source of the new methods that are developed.

Development
Thirdly, development consists of the construction of new methods of the method fragments through a process called method engineering (Brinkkemper, 1996). An improved method engineering approach is delivered.

The produced method, the GX WebEngineering Method has different route maps for the different project types, to provide situationality.

Evaluation
The fourth step is the evaluation step. The methods are evaluated by bringing them in practice in running projects at GX. This is included in this research in the form of a case study. Corresponding to the development step, knowledge obtained in this step can lead to an adaptation of the proposal.

Conclusion
The last step embodies the consolidation of the results and writing the conclusions. Also, the evaluating of the methods likely result a new problem awareness and suggestions for further research.

1.4. Results

The following deliverables are produced in this research project:

- a terminology list of the most used terms in the content management world;
- a literature study to existing development approaches, which resulted in a schematic overview of methods, model and techniques, grouped by publishing year;
- an improvement to the existing method engineering process;
- a meta-modeling technique, resulting in a process-data diagram, for method engineering is developed;
- an overview of characteristics of CMS-based web application implementations is provided. With these characteristics, one can categorize projects into standard or complex implementation situations; and
- a method for implementing CMS-based Web applications: the GX WebEngineering Method.

1.5. Research outline

In part I, the theoretical framework of this research project is described. Covered is scientific literature concerning web applications and content management; information systems development methods; and method engineering. Part II describes the business case at GX, consisting of GX, GX WebManager and the development process at GX.

In part III the process that is followed to develop the new method is described. Also, a preparation is made for part IV, in which the method base is filled. This is done by analyzing the existing methods and developing process-data diagrams.

Finally, in part V, the last step of the assembly-based situational method engineering process, as described in part III is described. This step resulted in a method, called GX WebEngineering Method. For the purpose of validation, the method was tested in a case study. After this conclusions were drawn, and finally a discussion on the conclusion and recommendation on future research is provided.

Part I: Theoretical Framework

In this part, the theoretical framework of this project is described. Covered is scientific literature concerning web applications and content management; information systems development methods; and method engineering.

In chapter 2, a terminology of the information systems field, concentrating on the content management and web application area is given. Chapter 3 provides an overview of existing information systems development approaches. Finally, in chapter 4, related work concerning method engineering is described.

2 Terminology

2.1. Introduction

As technology moves forward, several new concepts are being introduced in the field of information systems. Examples are: hypermedia, web applications, content management systems and web information systems. These terms are sometimes interchanged or misused. This chapter provides an overview of the most common terms in this research area. Moreover, a diagram is provided to set the newest developments on content management area in context.

In Figure 2-1, all relevant concepts are structured. It should be noted that this is just a part of the information systems world. The concepts are not complete and some may be overlapping. In the next sections al concepts are defined.

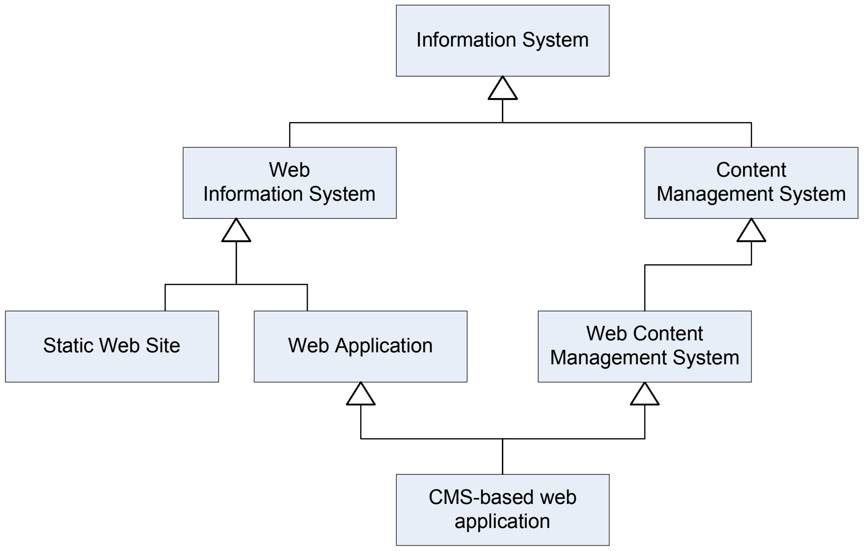

Information System
Web Information System
Content Management System
Static Web Site
Web Application
Web Content Management System
CMS-based web application

Figure 2-1: Terminology overview

2.2. Information systems

First of all, the term *information system* is often used. The various perspectives scientists have on the term information system are confusing. Hirschheim, Klein and Lyytinen (1995) for example, emphasized on the traditional view on information systems from two perspectives, namely the functional and structural perspective. Falkenberg et al. (1998) acknowledged the problem of terminology and constructed a framework of information system concepts. In accordance with Hirschheim et al. the Frisco Group recognized several interpretations, although these are not the same. It states that the term 'information system' is interpreted in at least three different ways: (a) as a technical system, implemented with computer and telecommunications; (b) as a social system, such as an organization in connection with its information; and (c) as a conceptual system (i.e., an abstraction of either of the above) (Falkenberg et al). To solve the problem of miscommunication between persons with different interpretations of information systems, Frisco Group developed the following definition:

An information system is a sub-system of an organizational system, comprising the conception of how the communication- and information–oriented aspects of an organization are composed (…) and how these operate, thus describing the (explicit and/or implicit) communication-oriented and information–providing actions and arrangements existing within that organization (Falkenberg et al.)

2.3. Web information systems

A special type of information systems is *web information systems*. According to Souer, Van de Weerd, Versendaal and Brinkkemper (2005), the many definitions that can be found of this concept all have one thing in common, namely, they all rely on the web for executing their program. In this work, their definition is adopted, namely: "Web information systems are a specialization of Information Systems which utilizes the technology of the Web" (Souer et al., 2005). This utilization of Web technology should be seen in a broad sense. A web information system could use the web solely for the purpose of presenting information. On the other hand, several web information systems use online applications for their interactive content.

2.3.1. Static websites

The first type of web information system is the *static website*. To define this type of web information system, a defection from DeTroyer and Leune (1998) is used. They aptly make a distinction between, what they define as, kiosk Web sites and application Web sites. They state that the first one mainly provides information and allow the user to navigate to that information. The latter one is described as an interactive information system where the user interface is formed by a set of Web pages (DeTroyer & Leune, 1998). In this work, the term static website is being used instead of kiosk web site. Hence, a static website provides the user with information and allows the user to navigate through that information.

2.3.2. Web applications

The second type of web information system is the *web application*. Gellersen and Gaedke (1999) define a web application as "any software application that depends on the World Wide Web, or simply the web, for its correct execution". Gnaho (2001) is more specific and defines a web application is as "an information system providing facilities to access complex data and interactive services via the Web which changes the state of business". In this definition the relation to information systems is clarified. Also, the abilities to access complex data and interactive services and change the state of business are typical for a web application.

In the existing literature sometimes other terms are used to address web applications. Michael Lang (2001), for example, uses the term *web-based hypermedia information systems*. This sounds logical, because Web applications can be seen as a combination of the processing power of information systems and the presentation and navigation facilities of hypermedia. However, according to Baresi, Garzotto, and Paolini (2000), Web applications are different from hypermedia for three main aspects: (a) users do not only navigate, but also activate operations and transactions; (b) the hypermedia structure itself may evolve as the application evolves; and (c) different users may have different visibility of the information and different capabilities for the operations.

In this work the term Web application, instead of web-based hypermedia information system, is used, which is defined by: "any software application that depends on the World Wide Web, or simply the Web, for its correct execution" (Gnaho, 2001). By using this definition, a clear distinction is made between Web applications and static Web sites. The term web application is also used to address to the end product that is developed with and based on WebManager.

2.4. Content management systems

A content management system (CMS) makes it possible to create, archive, search, control and publish information from within a flexible and integrated environment (Burzagli, Billi, Gabbanini, Graziani & Palchetti, 2004). Since the development of the web, CMSs are used for web purposes. Browning and Lowndes (2001) go a little further, by stating that the Web has become the preferred vehicle for content delivery, caused by its pervasive nature, and we should read CMS as web content management system. In this work, this distinction is acknowledged. Therefore, web content management system is listed as a separate concept.

2.4.1. Web content management systems

Vidgen, Goodwin and Barnes (2001) mention the following antecedents and enablers of web content management: document management, workflow integration, customer relationship management, e-commerce, software configuration management and data management. All these disciplines are covered in web content management.

McKeefer (2003) defines *web content management systems* as follows: "A WCM system consists of the software tool(s) used to provide the automated support of WCM activities", where web content management "incorporates the activities involved in the creation and deployment of digital content to Web based audiences, where these audiences may consist of customers, suppliers, partners and staff accessing the web content via extranet, internet, or intranet". This definition is adopted, but adjusted slightly. The part "accessing the web content via extranet, internet or intranet" is omitted. The reason for this is that not all web content management systems are web-based. Examples are Vignette and Microsoft CMS server, for which a software program needs to be installed. In this work a web content management systems is defined as "a system for the automated support of activities involved in the creation and deployment of digital content to web-based audiences, where these audiences may consist of customers, suppliers, partners or staff".

2.4.2. CMS-based web applications

A special type of web content management systems are CMS-based web applications. Souer, Van de Weerd, Versendaal and Brinkkemper (2005) define CMS-based web applications as "Web applications for the management and control of dynamic information". Examples are GX WebManager, FatWire Content Server and Tridion Web Content Management Edition.

CMS-based web applications are a specification of web content management systems. An important characteristic of the latter is that it needs the web for its correct execution. Web content management systems, however, are not by definition web-based.

2.5. Additional terms

The term *hypermedia* (and the variants *hypermedia system* and *hypermedia application*) is used many times in scientific literature, for example by Koch (2001). This term is often interchanged with the term *hypertext*. However, there is a difference. Hypertext is text, organized in a non-linear and non-sequential way, via nodes and links. Hypermedia is actually an extension of hypertext and combines hypertext and multimedia. Conklin (1987) defines hypermedia as "a style of building systems for organizing, structuring and accessing information around a network of multimedia nodes connected together by links". Nowadays, this term is often used to refer to web sites. This is confusing, since hypermedia is not necessarily Web-based. To avoid ambiguity, this term is not used in the rest of this work.

3 Existing development approaches

3.1. Introduction

Several development approaches (be it methods, models or techniques) for information systems and web applications exist. In this chapter, an overview of the main methods, models and techniques as described in scientific literature is presented.

3.2. Overview

To structure the vast amount, an overview is developed, to describe the relations between the different approaches. This overview is inspired by Lang's Evolution of Hypermedia Development Methods (Lang, 2002), as is depicted in Figure 3-1.

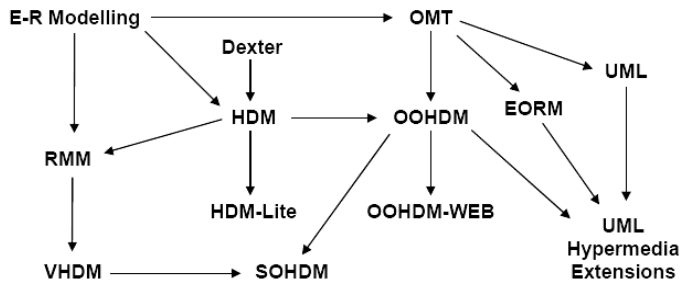


Figure 3-1: Evolution of Hypermedia Development Methods (Lang, 2002)

In Figure 3-2 the new overview is depicted. Main differences between the overviews in Figure 3-1 and Figure 3-2 are:

1. The model has been extended with several UML-based methods, models and extensions.
2. Some models which are not of great significance in research have been left out for the purpose of surveyability. This is also the case for the web and lite versions of certain methods.
3. The relation between E-R modeling and OMT is removed. In my opinion OMT did not evolve out of E-R-modeling, but was a completely new kind of modeling. Except for the timeframe and research field, no relationship exists.

In Figure 3-2 a new overview of methods, models, techniques and their influences on each other, is depicted. On the y-axis a global time indication is given.

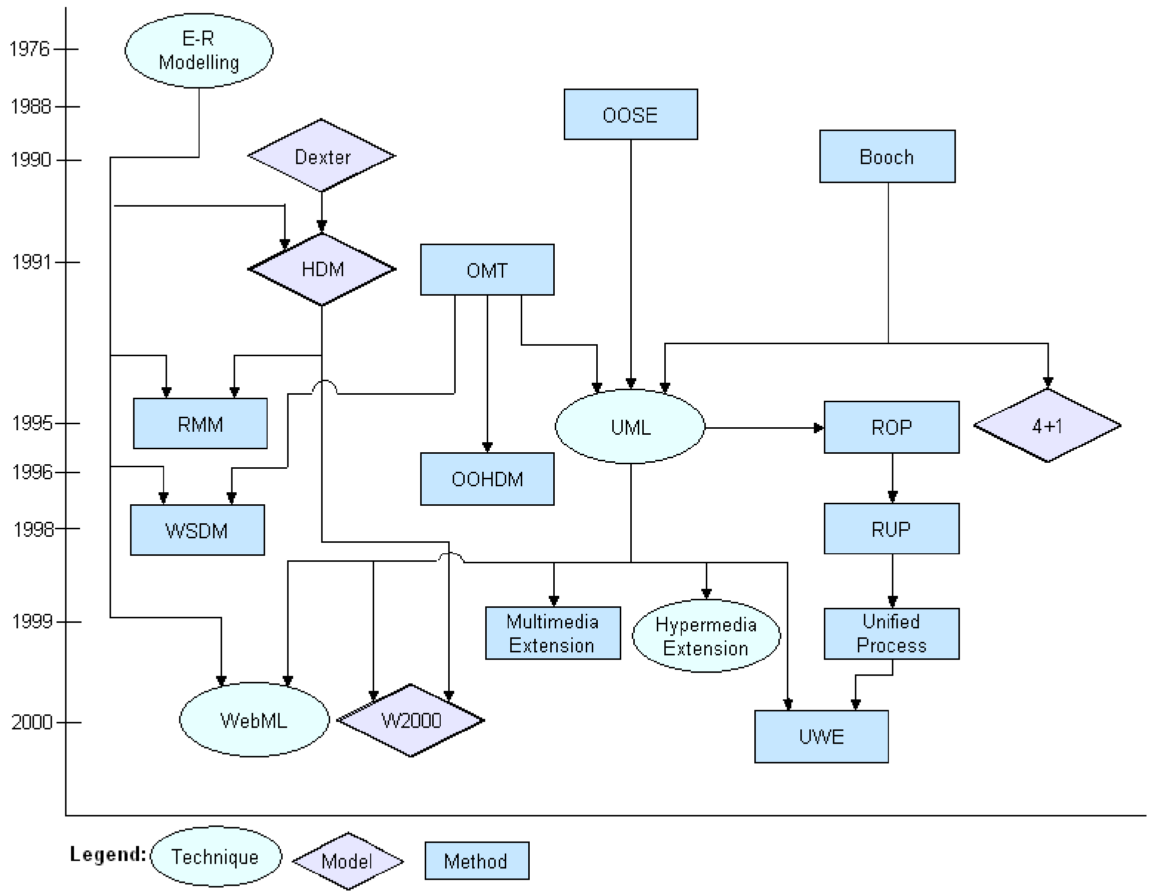


Figure 3-2: Relations between design methods, models and techniques

3.2.1. Entity-relationship modeling

Entity-relationship modeling, a technique for the database field, was proposed by Chen (1976). This technique describes conceptual data models on a high level in the form of entity-relationship diagrams, a specially developed notation for representing these models.

The next model is the Dexter Hypertext Reference Model which was developed by Halasz and Schwartz in 1990. The model divides hypertext into three layers, the run-time layer, the storage layer and the within component layer (Halasz and Schwartz, 1990).

The Dexter model received some criticism, which led to several contributions to improve the model. The Hypermedia Design Model (HDM) was proposed by Garzotto, Paolini and Schwabe in 1991. Fraternali (1999) states that: “HDM integrates features of the entity relationship model and the Dexter model, to obtain a notation for expressing the main abstractions of a hypermedia application, their internal structure and navigation, and application-wide navigation requirements”. Especially for web-based systems an adaptation of HDM was developed, which is called HDM-Lite (Fraternali & Paolini, 1998).

HDM is appropriate for describing the structure of the application domain (Isakowitz, Stohr & Balasubramanian, 1995). In 1995, Isakowitz, Stohr and Balasubramanian use the E-R modeling in their Relationship Management Methodology (RMM). This theory is based on the HDM, but provides a formal method in the process to design hypermedia applications. The method consists of seven steps from design through to construction. RMM however is not suitable for web applications. According to Conallen (1999), web applications are "business logic centric, and include a number of technological mechanisms for implementing business logic that are not adequately covered by RMM notation". Hence, the RMM notation is focused on relations and is hardly able to translate a system's business logic into specific elements and technologies.

3.2.2. Object-oriented modeling and UML

Rumbaugh started with object-oriented modeling in 1991 with the Object Modeling Technique (OMT) (Rumbaugh, Blaha, Premerlani, Eddy & Lorenson, 1991). This technique had some shortcomings, especially describing object interactions (Lang, 2001). However, OMT is at the base of several other methods. The Object-Oriented Hypermedia Design Model (OOHDM) by Scwhabe and Rossi (1995) is an extension of the HMD and based on OMT. The model encapsulates conceptual modeling, navigational design, abstract interface design and implementation. In the special environment OOHMD-Web OOHMD can be used to for creating hypermedia applications (Schwabe and de Almeia Pontes, 1998).

OMT is one of the techniques which concepts are integrated in the Unified Modeling Language (UML). UML is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system (Booch, Rumbaugh & Jacobson, 1999). Other techniques are OOSE and Booch. Jacobson, Christerson, Johnsson and Overgaard (1992) used use cases for the first time in 1992 to depict actions between the system and external entities. This Object-Oriented Software Engineering also had a commercial variant, called Objectory. In the Booch method the structure of a system is represented by class and state diagrams (Booch, 1990). The method covers the analysis and design phases of the development process.

The Booch notation was also used in the 4+1 View Model of Architecture developed by Philippe Kruchten of Rational Software. The model organizes a description of a software architecture using five concurrent views; four views to capture design decisions and the fifth view to illustrate and validate them (Kruchten, 1995). The views are: (a) the logical view to support the functional requirements; (b) the process view for requirements like performance and availability; (c) the development view for focusing on the organization of the software modules in the development environment; and (d) the physical view for requirements such as system availability and performance. The fifth view (e) is the scenario view, which makes it possible to show that the elements of the four views work together (Kruchten, 1995).

In 1995, a first version of UML was released by Booch, Rumbaugh and Jacobson. A lot of feedback was received from the business as well as the academic world. In 1997, the Object Management Group released the official version 1.0 of UML, integrating the techniques developed by Booch, Rumbaugh, and Jacobson.

UML was the beginning of a new era in the information systems field. The standard became widely used by commercial and non-commercial organizations. UML also was the basis of several specialized techniques for designing web applications and information systems.

3.2.3. The Unified Process

UML is graphical language, and should be used within the context of an end-to-end software process (Jacobson, Booch & Rumbaugh, 1999). The Rational Objectory Process was the first method that used UML. By the acquisitions and merging of several software companies, ROP received several contributions from different areas to improve the process. In 1998 a new version of the product was released: the Rational Unified Process, a use case driven iterative software engineering process, aimed at guiding software development organizations in their endeavors (Kruchten, 1999). Finally, in 1999 Booch et al. published the Unified Software Development Process, the follow-up of RUP, with as major change the unification in development approaches.

The Unified Process' distinguishing features are captured in the following key words: (a) use-case driven; (b) architecture centric; and (c) iterative and incremental. The lifecycle consists of 4 phases, namely inception, elaboration, construction and transition. Every phase ends with a milestone. A milestone means the deliverance of artifacts, models or documents that have been brought to a prescribed state (Jacobson, Booch & Rumbaugh, 1999). Each phase consists of several iterations. In the four phases five core workflows are addressed, which are: requirements, analysis, design, implantation and test.

The Unified Process is a very general process, but in RUP 5.5 (Rational Software Corp., 1999) a roadmap for developing web applications has been specified. The roadmap offers several extensions and variations on the activities of the regular process. One example is the introduction of a new artifact: the User-Interface Guidelines. In this document the graphic standards, user interactions, web techniques and related concepts are described.

3.2.4. Web modeling approaches

Since 1998 several methods and techniques for designing web applications have been developed. First of all, in 1998 the Website Design Method (WSDM) was developed by De Troyer and Leune. WSDM is a user-centered method for the design of kiosk Web Sites. A kiosk web site mainly provides information and allows users to navigate through that information (De Troyer & Leune, 1998). The two basic characteristics of WSDM are the audience driven approach, and the explicit conceptual design phase. The conceptual design can be performed by using techniques like OMT or E-R modeling. WSDM is a method specifically for developing kiosk web sites. Therefore, it is not suitable for developing complex web applications.

Secondly, Sauer and Engels proposed the UML Extension for Modeling Multimedia Applications. A multimedia application is an application that combines at least two media objects and shows time-dynamic behavior (Sauer & Engels, 1999). Aspects of the application that are covered in this extension are: (a) logical structure; (b) spatial presentation; (c) predefined temporal behavior; and (d) interactive control. Another extension was developed by Baumeister, Koch and Mandel (1999). They propose the UML Extension for Hypermedia Design, because the diagrams of UML are not sufficient to model aspects as navigational space and graphical representation.

In 2001 Koch proposed the UML-based Web Engineering approach. This approach is object-oriented, visualized with UML and based on the Unified Process. UWE covers the whole development process, which is divided in requirements analysis, conceptual, navigation and presentation design (Koch, 2001).

WebML is a notation for specifying complex web sites at the conceptual level (Ceri, Fraternali & Bongio, 2000). Its specification consists of four perspectives: (a) the structural model; (b) the hypertext model; (c) the presentation model; and (d) the personal model. It is not based on UML, but it is compliant with it. That is, it does not propose a new language for data modeling, but is compatible with existing notations as E-R modeling and UML. Also, WebML supports XML syntax, which can be used by software generators.

Finally, W2000 is a framework for designing web applications based on the preexisting assets UML and HDM (Baresi, Garzotto & Paolini, 2000). According to the authors, the integration between UML and HDM consists in:

1. Defining several stereotypes and customizations of diagrams to render HDM with UML.
2. Specifying guidelines to use UML as a way to specify some of the dynamic and operational aspects of web applications.
3. Refining use case diagrams to describe high-level user requirements, related to both functional and navigational aspects (Baresi et al., 2000).

4 Method engineering

4.1. Introduction

In this chapter, an overview is given on the research field of method engineering. Since the concepts used in method engineering sometimes are interchanged or misused, first a terminology is provided to structure the concepts used in the method engineering literature.

4.2. Terminology

Several terms are in use in the field of method engineering. In Figure 4-1 the relations between several important concepts are illustrated. In the following of this section these are further defined.

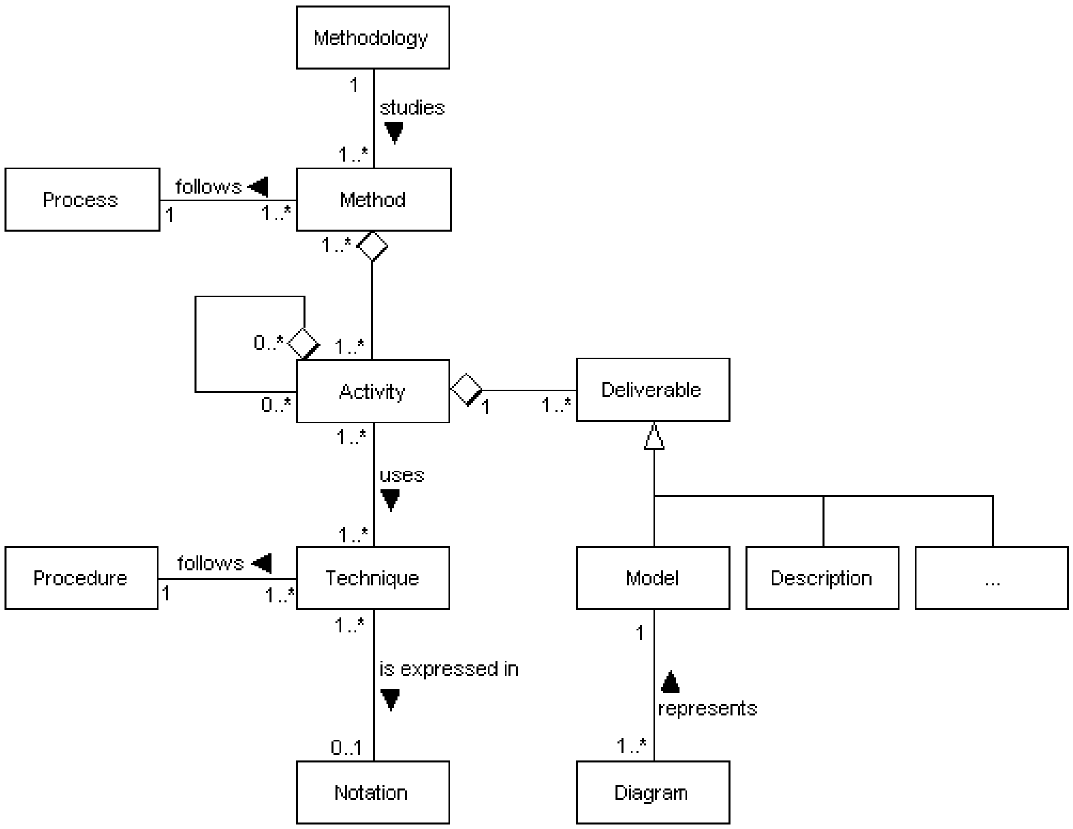


Figure 4-1: Meta-model of method engineering concepts

Brinkkemper (1996) defines a *method* as "an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products". This term differs from *methodology*, which refers to a system of methods in a particular discipline. For example, the methodology of information systems development is "the systematic description, explanation and evaluation of all aspects of methodical information systems development" (Brinkkemper, 1996).

A method can be seen as an organized set of techniques (Rossi & Brinkkemper, 1996). In the diagram the method is organized in *activities*, which use one or more techniques. Each activity produces a *deliverable*. In this meta-model two deliverables are depicted: *model* and *description*. This list is not complete, these are merely examples.

A *technique* is a procedure, possibly with a prescribed notation, to perform a development activity (Brinkkemper, 1996). A *procedure* describes how specific tasks have to be carried out and the term *notation* refers to the graphical language that is used to express the method. The terms language and notation are used interchangeably in this research.

According to Booch, Rumbaugh and Jacobson (1999) a *model* is "a simplification of reality, in which reality is defined in the context of the system being modeled". A model is represented by one or more diagrams. In the UML User Guide a *diagram* is described as "a graphical representation of a set of elements" (Garzotto, Paoline & Schwabe, 1991). Complex models can be represented by a number of diagrams, each with its own view. In Figure 4-1 is illustrated that a model is represented by a diagram.

The concepts *process* and *procedure* are often mixed up. Some use the terms synonymous, others use it with a different meaning. In the FAQ of the ISO 9000 Standard these terms are defined as follows: "A process may be explained as a set of interacting or interrelated activities, which are employed to add value. A procedure is a method of describing the way in which all or part of that process is to be performed" (Organization of Standardization, 2004). In this project the term process is used in a general form, for example in the 'Unified Process'. The term procedure is used to refer to the specific algorithm that is used when one is, for example, drawing a class diagram.

4.3. Literature overview

4.3.1. Method engineering

Kumar and Welke (1992) state that "there is no detailed information systems methodology which is the best in all situations". They introduced a solution to this problem, method(ology) engineering, which describes the engineering of information systems development methods, by taking into account the uniqueness of a project situation. In addition, Brinkkemper (1996) defined method engineering as "the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems".

4.3.2. Situational method engineering

A special type of method engineering is situational method engineering. Kumar and Welke (1992) stress the importance of distinguishing development situations. The term situational method is defined as "an information systems development method tuned to the situation of the project at hand" (Harmsen, Brinkkemper & Oei, 1994).

Situational method engineering is often used in combination with route maps, which were introduced by Van Slooten and Brinkkemper (1993). Route maps can be used to tune the method into situational methods (Van Slooten & Hodes, 1996; Aydin & Harmsen, 2002). Different routes are used to represent the different situations,

4.3.3. Assembly-based situational method engineering

Recent research in the method engineering area is done by Ralyté, Deneckère and Rolland (2003). They developed a generic process model for situational method engineering. This process model contains three approaches: (a) the assembly-based strategy, based on the reuse of method components extracted from existing methods and stored in some method base; (b) the extension-based strategy, used for extending a method by applying extension patterns; and (c) the paradigm-based strategy, when a new fresh method must be constructed either by abstracting from a given model or by instantiating a meta-model (Ralyté, Deneckère and Rolland, 2003).

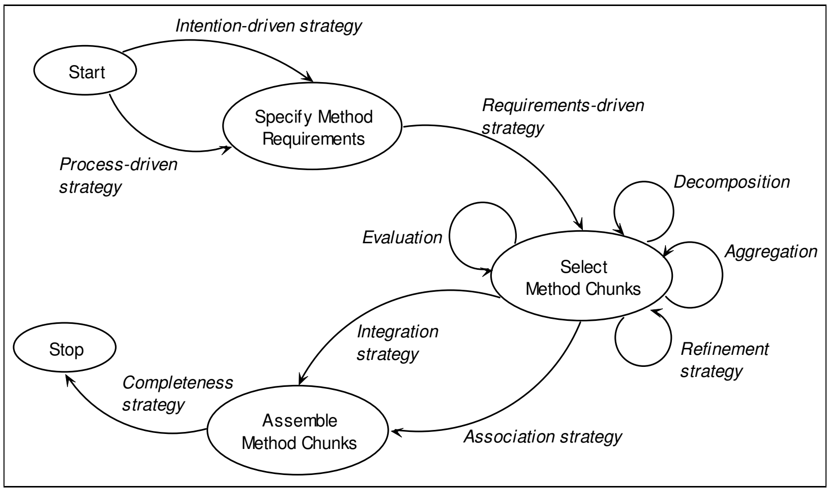


Figure 4-2: Assembly-Based Process Model for Situational Method Engineering

In Figure 4-2 the assembly-based process model for situational method engineering is illustrated. Three key intentions are represented: specify method requirements, select method chunks and assemble method chunks. This approach is used in this research and will be elaborated on in the next chapter.

4.3.4. Method configuration

To be complete, another type of method engineering is outlined in this section. Karlsson (2002); and Karlsson and Ågerfalk (2004) use method configuration to adapt a particular method to various situated factors. The difference with assembly-based method engineering is that the focus is on one method which is configured in a particular situation, instead of using a set of methods as a base for assembly. Hence, method configuration is, just like assembly-based method engineering, treated as a particular kind of method engineering.

Part II: Case

In this part, the case at GX is described, consisting of GX, GX WebManager and the development process at GX. In this business case, the method is developed, tailored and tested

In chapter 5, GX creative online development and GX WebManager are described. Subsequently, in chapter 6, the existing development process at GX is described.

5 GX creative online development

5.1. Company

GX is a web technology company, active in the fields of content and web management, online application development, and integration of backend systems in portal solutions. In 1995 the foundation *GX Group* was founded and in 1998 it changed its name in *GX creative online development B.V.* Currently, about 70 employees are working at GX.

The industries GX works in are diverse; they comprise services, sports organizations, publishing companies, media, government, education, knowledge centers and health care. Important accounts are: Ajax, DaimlerChrysler, KPN, Asics, Levob, Planet Internet, Unicef, Voetbal International, VNO-NCW, Volkskrant, Talpa, several small and large municipalities.

5.2. GX WebManager

GX implements web applications, using GX WebManager, a generic CMS-based web application. GX WebManager enables people without a specific technological background in creating, maintaining and integrating several dynamic websites and portals.

GX WebManager consists of several components: framework management, layout and presentation management, content management, interaction management, connectivity management and workflow management. Customers can choose which components they want to purchase. The WebManager architecture is depicted in Figure 5-1. More information on GX WebManager can be found in Van Berkum, Brinkkemper and Meyer (2004).

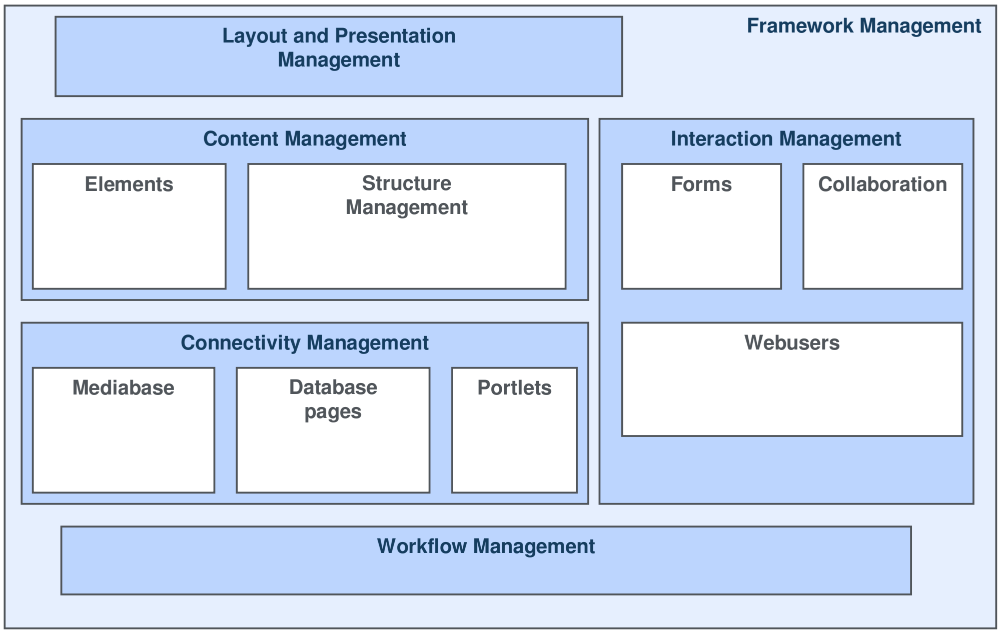


Figure 5-1: GX WebManager architecture

6 Development process at GX

6.1. Introduction

In the first section in this chapter the processes and workflows at GX are described. Secondly, the scope of this research is indicated. Finally, an inventory of standard methods and techniques used at GX is provided.

6.2. Processes & workflows

Figure 6-1 shows the activities that are carried out during a project. Different processes are depicted: Accounting, project phases, workflows and project management. Also, several project products and other deliverables are described in the figure.

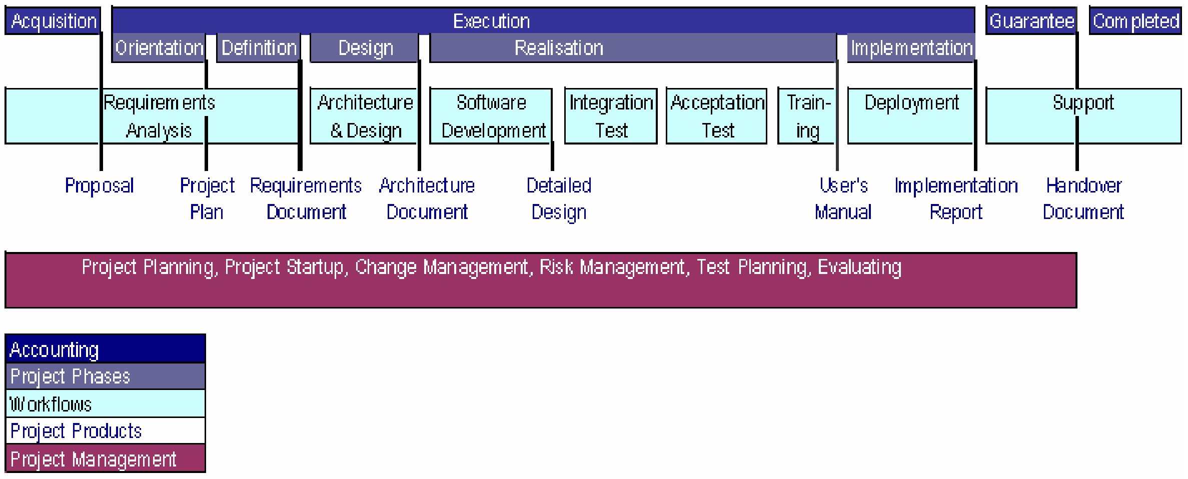


Figure 6-1 – Processes & workflows at GX

The illustrated processes all contribute to one end product, the web application. Every process is on a different level and has its own view. In the next sections the characteristics of each view are explained.

6.2.1. Accounting

On the highest level the accounting stages are illustrated. The accounting process consists of four stages: (a) the acquisition stage is the period until the customer approves the proposal; (b) the execution stage is divided into five project phases which will be covered in the next section; (c) the guarantee stage starts after the project acceptation by the customer; and (d) completed, the last stage, starts when the guarantee ends.

6.2.2. Project phases

The execution stage of the accounting process is divided into five project phases:

- Orientation phase: During this phase the project plan is developed. Resources are assigned and a planning is developed.
- Definition phase: In this phase is defined what to build. The phase results in a requirements document.
- Design phase: Here is defined how to build the web application. The architecture document is the deliverable of this phase.

- Realization phase: In this phase the actual web application is being built. At the end of the phase a working en tested web application is delivered.
- Implementation phase: During this phase the web application is implemented and the implementation report is delivered.

6.2.3. Workflows

Several workflows are depicted in Figure 6-1. First of all, in the *requirements analysis* a description is written of the functional and non-functional requirements of the web application. This document is written by a consultant who collects information based on two sources of information. Generally, an account manager collects the wishes of the customer and hands this over to the consultant. Transferring the information happens by means of arranging a short pre-sales intake and handing over the relevant documents. At the end of the acquisition phase, a proposal is produced. This proposal consists of a work-break-down and a written approach for the solution. Also the quotation is formulated and has to be signed by the customer. At the end of the definition phase a first version of the requirements document is delivered.

Secondly, in the *architecture & design workflow,* the requirements are translated into a realizable solution. For complex projects an architectural design per component and a technical design is delivered.

Next, in the s*oftware development workflow* the actual realization of the web application is performed. Subsequently, this detailed design is first tested in an *integration test*, and finally in the *acceptation test*, after which the client has to give his approval. Also the user's manual is written in this period.

When the client accepts the product, the *training* can start. Editors of the concerning company are taught how to use WebManager. Although this is modeled at the end of the realization phase, it sometimes occurs that training already is given earlier in the process. The customer determines the timing of the training.

After the training the *deployment* starts. The website is transferred to the final production environment and the site can be filled with content. The deployment workflow results in a fully operational web application and an implementation report.

After the acceptation of the website, the *support flow* starts where the project manager hands over the project. From this moment on all problems, comments and new wishes are handled via the support department.

One workflow is not represented in the figure: the *consultancy workflow*. This is because this workflow is difficult to classify in one project phase, since consultancy consists of several activities in different phases. This workflow is clarified in another section of this chapter.

6.2.4. Project management

Project Management at GX is based on the PRINCE II method. Several activities exist, encompassing: (a) planning, which results in a project plan; (b) startup, with typical activities as meeting with the graphical design party, prepare a hosting environment, contacting other third parties, and allocating resources; (c) change management, with the accompanying change requests; (d) risk management, resulting in risk logs that are included in the progress report; (e) test planning, for testing the product; and (f) evaluation, which is documented in an evaluation report.

6.3. Scope of this research

In Figure 6-1 the main processes of GX are depicted. Several flows in this figure are not relevant in this report. The problem definition of this research project is “How should a design method be constructed for the process of developing web applications for GX WebManager?”. Accounting is not relevant for this question, since it is only used for accountancy objectives, and the Project Management flow doesn’t handle the design process, but the management processes. Therefore, the scope is limited to the project phases and workflows. During this research, however, one exception was found. The acquisition phase appeared to be very important for the rest of the process, since in this phase, the first requirements are being captured.

In Figure 6-1 the workflows are represented in sequential order. In reality these workflows overlap each other, as is illustrated in Figure 6-2. On the vertical axis the workflows are listed and on the horizontal axis the project phases are represented. In the figure also the consultancy flow is illustrated. Consultancy covers almost the whole process, and can be divided into three activities. First, there is the pre-sales consultancy, secondly the requirements analysis consultancy and finally the implementation consultancy.

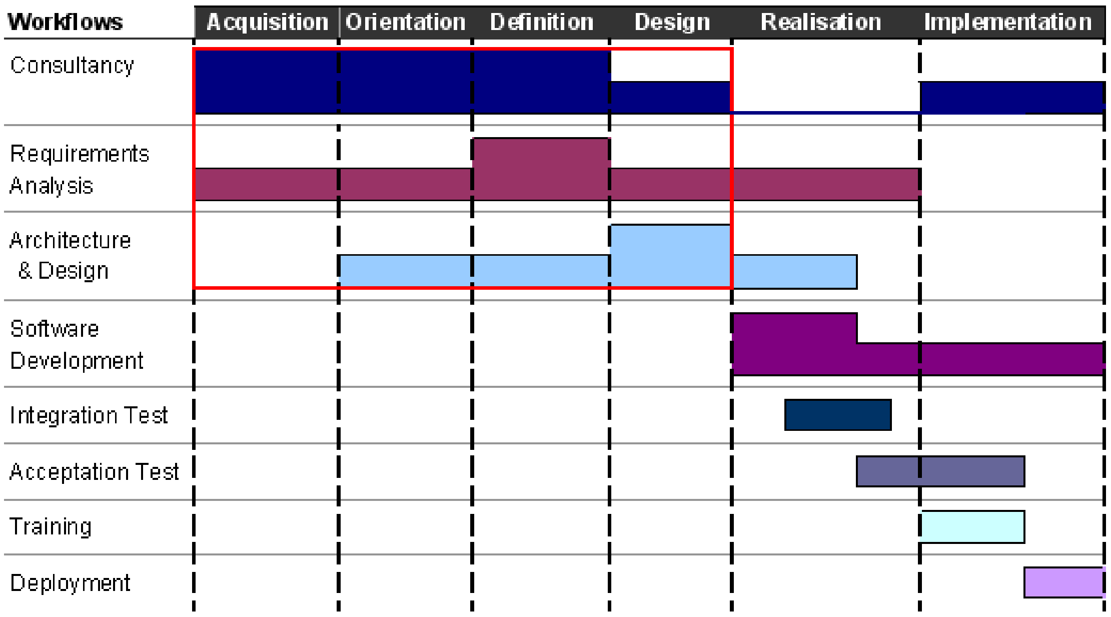


Figure 6-2: Scope of project phases and workflows

The scope of this research project is marked with a red frame. Only the acquisition, orientation, definition and design phases are covered, which implies that only the workflows consultancy, requirements analysis and architecture and design are part of this research.

6.4. Standard techniques used at GX

In this section an overview of standard methods, models and techniques used at GX is presented. The methods, models and techniques encountered during this research are based on the *Rational Unified Process*, the *4+1 View Model of Architecture* and the *IEEE Standard for Software Specification*. In the next sections a short description of every concept is given and shown how it is implemented at GX.

6.4.1. Rational Unified Process 5.5

In the requirements analysis parts of the Rational Unified Process 5.5 are adopted. This is not the case in all projects, but only the larger ones. The layout of the requirements analysis document is based on the Software Requirements Specification as described in RUP 5.5 (Rational Software Corp., 1999). Use cases are used to express the functional requirements of the website. For each use case, or use case subset, a use case description is given. Also, a supplementary specifications template is used in combination with the IEEE standard for Software Specification to describe the additional software requirements.

6.4.2. 4+1 View Model of Architecture in RUP 5.5

During the design phase an architecture document is written. To describe the architecture, the software architecture document template of RUP 5.5 (Rational Software Corp., 1999) is used. In this template, the 4+1 View Model of Architecture is adopted to describe the architecture of a system. The 4+1 View Model of Architecture organizes a description of a software architecture using five concurrent views; four views to capture design decisions and the fifth view to illustrate and validate them (Kruchten, 1995). In RUP 5.5 six views are described, which are: use-case view, logical view, process view, deployment view, implementation view and data view. GX adopted 4 of these views (logical view, process view, deployment view and implementation view) and added the requirements view, as is depicted in Figure 6-3.

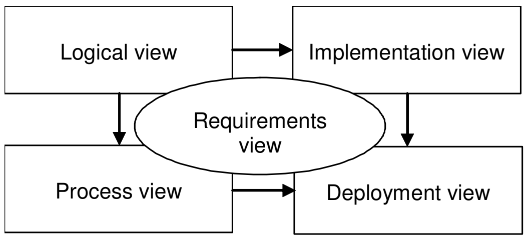


Figure 6-3: 4+1 View Model of Architecture (Rational Software Corp., 1999)

In the following, a description of the 4+1 View Model of Architecture is given. First, the logical view is depicted. This view is used to describe the structure of the web application by describing the different functional components. The second view is the process view, where the system is described in terms of processes en communication between these processes. Important issues are performance, scalability and throughput capacity. In the third view, the implementation view, the architecture is described from a software development perspective. The fourth view is the deployment view which contains a description of the physical nodes for typical platform configurations and the allocation of tasks (as described in the process view) in the physical machine. The *+1 view* is the requirements view. In this view the requirements and technical risks which can be important for the architecture are described. This view is used to verify the other views.

6.4.3. IEEE Standard for software specification

In the requirements specification of large projects a section with additional requirements are included. These requirements are non-functional and can be typed as quality characteristics, because they provide boundary conditions that have to be satisfied by the functional requirement. The additional requirements are divided into the following categories: usability, supportability, reliability, performance, scalability, interface, additional functionality, and design constraints (IEEE, 1998).

Non-functional requirements are difficult to measure. To choose which requirements are most important, and thus should be implemented, the priorities have to be determined. Two factors are used to determine this: desirability and impact. Desirability indicates the importance of the requirement and is the most significant factor. When the additional requirements are equally desirable, the requirements are prioritized on the impact score. The impact score indicates the consequences for the system when the additional requirement concerned is not implemented. When the impact is disastrous it means that it should be implemented, because without it the system does not function properly anymore. For every additional requirement the title, description, desirability, impact and related use cases have to be listed.

PART III: Assembly-based Situational Method Engineering

In this part the process that is used to develop the new method is described. Also, a preparation is made for part IV, in which the method base is filled.

In chapter 7, the approach of assembly-based situational method engineering is outlined. Then, in chapter 8, the meta-modeling technique that is used to analyze and assemble the methods is described. In chapter 9 projects characteristics are analyzed and used to make a categorization of implementation situations. Finally, in chapter 10, is described which candidate methods are used from the scientific literature to fill the method base.

7 Method engineering approach

7.1. Introduction

In this research project, method engineering is used to develop new methods. Rossi, Tolvanen, Ramesh, Lyytinen and Kaipala (2000) mention the development of UML extensions as a reaction to the abundance of variants of UML for special purposes as a prime example of successful situational method engineering. Also Dietzch (2002) showed that situational method engineering could be used as an appropriate approach for solving the problem to finding the right method.

In this chapter, the method engineering approach that is used in this project is described. Also, the meta-modeling technique supporting the method engineering approach is described.

7.2. Assembly-based situational method engineering

The approach to situational method engineering described in most literature is quite clear. Brinkkemper (1996) recognized the following steps: (1) characterization of the project, (2) selection of method fragments (that are stored in a method base), and (3) assembly of method fragments. The experience gained in this process is new input for the method base. Saeki (2001) states that the simplest way to construct a new method is first to put meaningful method fragments in a method base, then to select useful method fragments from this method base, and finally adapt and integrate them in a new method. Ralyté, Deneckère and Rolland (2003) have developed the assembly-based process model for situational method engineering. This model describes three steps to develop a new situational method. The steps are: (1) specify method requirements, (2) select method fragments and (3) assemble method fragments.

In the described research it is either assumed that the method base with method fragments is already filled, or that the methods that are to be stored in the method base are already selected. In case of developing methods for a relatively new information systems field, in this case CMS-based Web applications, the method base needs to be filled first. Therefore, the following steps are proposed to develop a new method for implementing CMS-based web applications, by means of assembly-based situational method engineering:

1. Analyze implementation situations and identify needs.
2. Select candidate methods that meet one or more aspects of the identified needs.
3. Analyze candidate methods and store relevant method fragments in a method base.
4. Assemble a new method from useful method fragments and use route map configuration to obtain situational methods.

The third and fourth steps are supported by a meta-modeling technique, especially developed for method engineering purposes. This technique, in which a process-data diagram is built, is used in analyzing, storing, selecting and assembling the method fragments.

In the next sections, the first two steps of the method engineering process are described. Then, in chapter 8, the meta-modeling technique that is used to support the selection and assembly process is outlined.

The third step, filling the method base is described in part IV. Finally, the last step, where the actual method is developed, is described in part V. In the method rationale is described where the method fragments originate from and why they are chosen. After that, the process-data diagrams of the method are illustrated.

7.3. *Implementation situations*

The first step in the method engineering process is the implementation situation identification. For different situations, situational methods will be developed, expressed in the route map of the new method.

7.3.1. Projects

In 2004 GX completed 80 implementations, which vary in size, sector and type. The number of employees of the client organizations ranges from a few to tens of thousands of employees. Several types of implementations exist, for example the creation of a new website from scratch, or the migration of an existing website to the WebManager system. In the next section a categorization is presented for these implementations.

7.3.2. Categorization

To develop situational methods for WebManager implementations, the existing implementation situations need to be defined first. This categorization is done by means of artifact analysis and semi-structured interviews with project managers. Artifact analysis comprises the research in the documentation of guidelines and finished projects. After making a first categorization, this is discussed during the semi-structured interviews. Subsequently, a final categorization is developed.

Resulting from artifact analysis and semi-structured interviews, three kinds of implementation situations are identified: standard, complex and migration implementation situations. The latter one was easy to identify. When a large amount of content from an existing web application needs to be migrated to the new Web application, this is classified as a migration implementation situation. However, the difference between standard en complex implementation situations is more ambiguous. The solution to this problem is found in the existing method engineering literature.

Kumar and Welke (1992) as well as Brinkkemper (1996) stress the importance of distinguishing implementation situations. In this research, the term *implementation situation* is used, since the project deals with the implementation of a CMS-based Web application. The categorization of implementation situations is based on their distinguishing characteristics. Karlsson (2002) followed a similar process in abstracting projects into implementation situations for the purpose of method configuration. He defined a characteristic of an implementation situation as: “a delimited part of an implementation situation, focusing on a certain problem or aspect which the method configuration aims to solve or handle”. This definition is used to define a characteristic of an implementation situation, that is: “a characteristic is a delimited part of an implementation situation, focusing on a certain problem or aspect which the method aims to solve or handle”.

Kumar and Welke (1992); and Van Slooten en Hodes (1996) mention several characteristics for the categorization of development projects that are of importance. In general one can state that these factors are deduced from the context, organization or from technical aspects from the project (Karlsson, 2002). In Table 7-1 the adopted characteristics per area are described.

Table 7-1: Implementation situation characteristics

Context	Dependency (to external activities & conditions)
	Level of innovation (of the applied technology, methods, tools and techniques)
Organization	Number of stakeholders
	Uncertainty of customer's expectations by management team
	Uncertainty of development activities by customer
Technique	Complexity (of functional components)
	Number of relationships (to existing systems)

The characteristics listed in Table 7-1 can be used to categorize the implementation situations. Every characteristic can be labeled with a value: high or low. In general one can state that the complexity of an implementation situation depends on the amount of characteristics that is labeled with a high value. When three or more of the values are high, it should be categorized as a complex implementation situation. Otherwise, it is a standard implementation situation.

Summarizing, three implementation situations are identified:

- Standard implementation situation
  Types of project that score less than three high values on the implementation characteristics are considered to be standard implementation situations.
- Complex implementation situation
  Types of project that score three or more high values on the implementation characteristics are considered to be standard implementation situations. Excluded from this situation are updates of existing WebManager web applications. Included are migrations from existing non-WebManager web applications to WebManager web applications.
- Migration implementation situation
  Migrations occur when a client wants to migrate its existing web application to the WebManager system. Two types of migration exist:
  - migration from an old version of WebManager to a newer version; and
  - migration of the content of an existing web application to WebManager.

  The latter is handled as a complex implementation situation. Hence, migration in this research will only comprise the update from an old version to a new version of WebManager.

7.3.3. Implementation situation needs

In this section, the main implementation situation needs are inventorised. These were obtained by conducting semi-structured interviews with consultants, project managers and software architects; and artifact analysis of existing requirements specifications and project evaluation documents. Several problems were found and translated into overall needs, standard implementation situation needs and complex implementation situation needs. In Table 7-2 needs are given for each implementation situation.

Table 7-2: Implementation situation needs

Overall needs	▪ The method should deliver a requirements document that is understandable for the customer and informative for the stakeholders at GX.

Standard implementation situation needs	▪ Standard project often have a small budget. This implies that the amount of time for specifying the requirements is limited. Therefore, the method should make it possible to translate the requirements quickly into WebManager solutions. ▪ Communication between pre-sales consultant and account manager should be improved, since often the account manager is the pre-sales consultant's direct link with the customer. ▪ Often, different types of project documents are delivered per project. Therefore, it is not clear which information can be found where. This should be clarified.
Complex implementation situation needs	▪ A solution has to be found to the problem of changing requirements after the contract is signed. Although one can expect the requirements to change during the requirements analysis, the customer often does not understand that this affects the budget. ▪ For modeling complex functionalities use cases are preferred. However, developing use cases is time demanding. Therefore, the method should offer the ability to develop use cases fast and reuse the knowledge. ▪ There should be a clear distinction between functional requirements and architectural specifications.
Migration implementation situation needs	▪ Customer's expectations should be right after signing the contract. ▪ It should be clear when to use a migration script and when to migrate manually. ▪ Implementation should only start when the architecture document is ready, (if used) the migration script is tested, and the code is reviewed. ▪ A document comparable to a requirements document should be written, in order to clarify the necessary activities. ▪ The method should have space for a thorough legacy system scan, since old WebManager installations are often badly documented.

7.4. Candidate method selection

The second step in the method engineering process is the selection of candidate methods from which method fragments are extracted and stored in a method base. The chosen methods are analyzed in part IV, the method base.

After conducting a literature research, the choice has been made to use the Unified Process (Jacobson, Booch & Rumbaugh, 1999) and UML-based Web Engineering (Koch, 2001) as candidate methods.

In choosing the candidate methods, the following considerations were taken into account: (a) the Unified Process is very suitable to divide into fragments and store in a method base; (b) UWE combines the strengths of the Unified Process with several Web-specific characteristics; (c) the Unified Process is a popular de facto standard modern software development process (Larman, Kruchten and Bittner, 2001), and known by the consultants who are going to use the method; and (d) both methods use UML as modeling language, which is the standard notation for modeling object-oriented systems and widely accepted by the software engineering community (Baresi, Gazotto and Paoline (2000).

8 Meta-modeling technique

8.1. Introduction

The technique used to model the activities and artifacts in the development process is a meta-modeling technique, expressed in process-data diagrams. Saeki (2003) proposed the use of this meta-modeling technique for the purpose of attaching semantic information to the artifacts and for measuring their quality using this information. In this research the modeling technique is adopted to reveal the relations between activities (the process) and artifacts (the data produced in the process).

A process-data diagram consists of two integrated diagrams. The left-hand side of the diagram is based on a UML activity diagram, and the right-hand side of the diagram is based on a UML class diagram. In this chapter first the left-hand side of the diagram is explained, then the right-hand side, and finally the integration of both diagrams.

In the following of this chapter, the meta-modeling technique is explained. Every theoretical explanation will be illustrated by an example in practice. This can be a fragment from the process-data diagram of the Unified Process, UML-based Web Engineering or the process at GX.

8.2. Meta-process modeling

Meta-process modeling is done by adapting the UML *activity diagram*. According to Booch, Jacobson and Rumbaugh (1999), an activity diagram is “a diagram that shows the flow from activity to activity; activity diagrams address the dynamic view of a system”. This diagram consists of activities and transitions. If necessary, activities can be divided into sub-activities. Transitions can be used to show the path from one activity to the next. A simple arrow depicts this. Four types of activities exist: unordered, sequential, concurrent and conditional activities.

8.2.1. Sequential activities

Sequential activities are activities that need to be carried out in a pre-defined order. The activities are connected with an arrow, implying that they have to be followed in that sequence. Both activities and sub-activities can be modeled in a sequential way. In Figure 8-1 an activity diagram is illustrated with one activity and two sequential sub-activities. A special kind of sequential activities are the start and stop states, which are also illustrated in Figure 8-1.

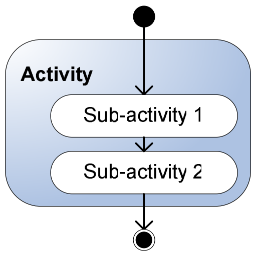


Figure 8-1: Sequential activities

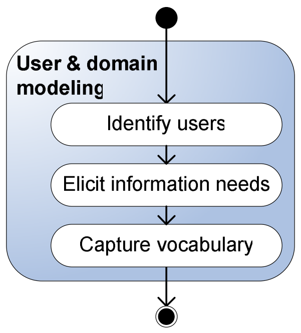


Figure 8-2: Example sequential activities

In Figure 8-2 an example from practice is illustrated. The example is taken from the requirements capturing workflow in UML-based Web Engineering. The main activity, user & domain modeling, consists of three activities that need to be carried out in a predefined order

8.2.2. Unordered activities

Unordered activities are used when sub-activities of an activity do not have a pre-defined sequence in which they need to be carried out. Only sub-activities can be unordered. Unordered activities are represented as sub-activities without transitions within an activity, as is represented in Figure 8-3.

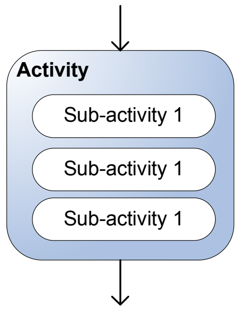


Figure 8-3: Unordered activities

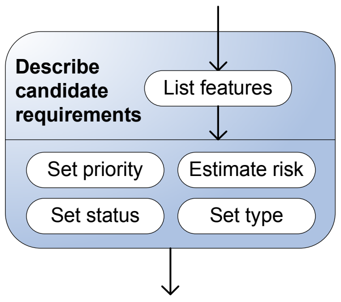


Figure 8-4: Example of unordered activities

In some specific cases an activity exists of sequential and unordered activities. The solution to this modeling issue is to divide the main activity in different parts. In Figure 8-4 an example is illustrated, which clarifies the necessity to be able to model unordered activities. The example is taken from the requirements analysis workflow of the Unified Process. The main activity, “describe candidate requirements”, is divided into two parts. The first part is a sequential activity. The second part consists of four activities that do not need any sequence in order to be carried out correctly.

8.2.3. Concurrent activities

Activities can occur concurrently. This is handled with forking and joining. By drawing the activities parallel in the diagram, connected with a synchronization bar, one can fork several activities. Later on these concurrent activities can join again by using the same synchronization bar. Both activities and sub-activities van occur concurrently. In the example of Figure 8-5 Activity 2 and Activity 3 are concurrent activities.

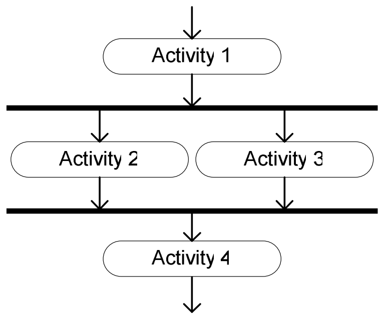


Figure 8-5: Concurrent activities

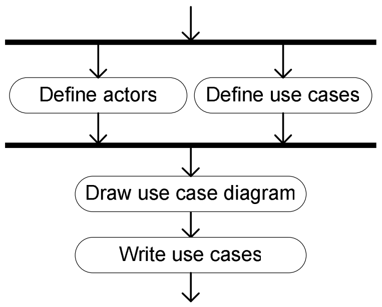


Figure 8-6: Example concurrent activities

In Figure 8-6 a fragment of the requirements capturing process at GX is depicted. Two activities, defining the actors and defining the use cases, are carried out concurrently. The reason for carrying out these activities concurrently is that defining the actors and the use cases influences each other to a high extend.

8.2.4. Conditional activities

Conditional activities are activities that are only carried out if a pre-defined condition is met. This is graphically represented by using a branch. Branches are illustrated with a diamond and can have incoming and outgoing transitions. Every outgoing transition has a guard expression, the condition. This guard expression is actually a Boolean expression, used to make a choice which direction to go. Both activities and sub-activities can be modeled as conditional activities. In Figure 8-7 two conditional activities are illustrated.

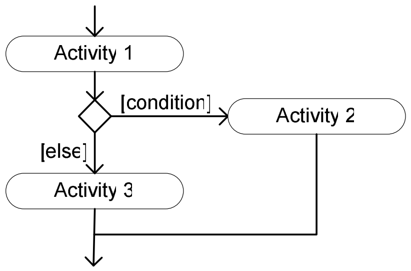


Figure 8-7: Conditional activities

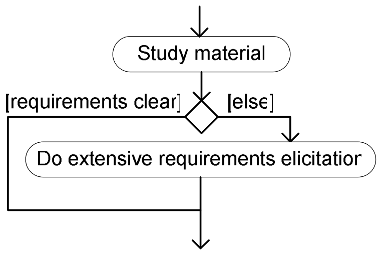


Figure 8-8: Example conditional activities

In Figure 8-8 an example from practice is illustrated. A requirements analysis at GX starts with studying the material. Based on this study, the decision is taken whether to do an extensive requirements elicitation session or not. The condition for not carrying out this requirements session is represented at the left of the branch, namely [requirements clear]. If this condition is not met, [else], the other arrow is followed.

8.3. Meta-data modeling

The meta-data side of the diagram consists of a *concept diagram*. This is basically an adjusted class diagram as described Booch, Rumbaugh and Jacobson (1999). Important notions are concept, generalization, association, multiplicity and aggregation.

8.3.1. Concepts

First of all, a concept is a simple version of a UML class. The class definition of Booch, Rumbaugh and Jacobson (1999) is adopted to define a concept, namely: a set of objects that share the same attributes, operations, relations, and semantics.

The following concept types are specified:

- STANDARD CONCEPT: a concept that contains no further (sub) concepts. A standard concept is visualized with a rectangle.
- COMPLEX CONCEPT: a concept that consists of a collection of (sub) concepts. Complex concepts are divided into:
- OPEN CONCEPT: a complex concept whose (sub) concepts are expanded. An open concept is visualized with two white rectangles above each other.
- CLOSED CONCEPT: a complex concept whose (sub) concepts are not expanded since it is not relevant in the specific context. A closed concept is visualized by a white rectangle above a black rectangle.

In Figure 8-9 the three concept types that are used in the modeling technique are illustrated. Concepts are always capitalized, not only in the diagram, but also when referring to them outside the diagram.

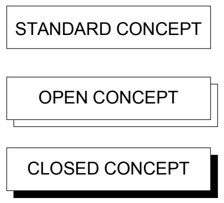


Figure 8-9: STANDARD, OPEN and CLOSED CONCEPTS

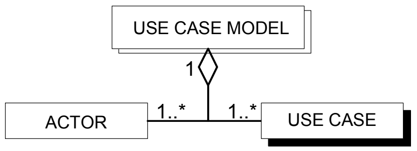


Figure 8-10: Example of STANDARD, OPEN and CLOSED CONCEPTS

In Figure 8-10 all three concept types are exemplified. Part of the process-data diagram of the requirements workflow in the Unified Process is illustrated. The USE CASE MODEL is an open concept and consists of one or more ACTORS and one or more USE CASES. ACTOR is a standard concept, it contains no further sub-concepts. USE CASE, however, is a closed concept. A USE CASE consists of a description, a flow of events, conditions, special requirements, etc. Because in this case we decided it is unnecessary to reveal that information, the USE CASE is illustrated with a closed concept.

8.3.2. Generalization

Generalization is a way to express a relationship between a general concept and a more specific concept. Also, if necessary, one can indicate whether the groups of concepts that are identified are overlapping or disjoint, complete or incomplete. Generalization is visualized by a solid arrow with an open arrowhead, pointing to the parent, as is illustrated in Figure 8-11.

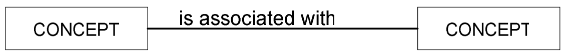


Figure 8-11: Generalization

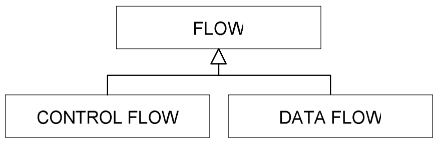


Figure 8-12: Example generalization

In Figure 8-12 generalization is exemplified by showing the relationships between the different concepts described in the preceding paragraph. STANDARD CONCEPT and COMPLEX CONCEPT are both a specific kind of CONCEPT. Subsequently, a COMPLEX CONCEPT can be specified into an OPEN CONCEPT and a CLOSED CONCEPT.

8.3.3. Association

An association is a structural relationship that specifies how concepts are connected to another. It can connect two concepts (binary association) or more than two concepts (n-ary association). An association is represented with an undirected solid line. To give a meaning to the association, a name and name direction can be provided. The name is in the form of an active verb and the name direction is represented by a triangle that points in the direction one needs to read. Association with a name and name direction is visualized in Figure 8-13.

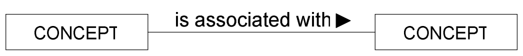


Figure 8-13: Association

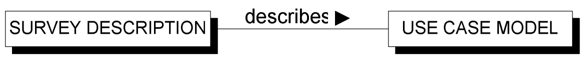


Figure 8-14: Example association

In Figure 8-14 an example of association is illustrated. The example is a fragment of the process-data diagram of the requirements analysis in the Unified Process. Because both concepts are not expanded any further, although several sub concepts exist, the concepts are illustrated as closed concepts. The figure reads as "SURVEY DESCRIPTION describes USE CASE MODEL".

8.3.4. Multiplicity

Except name and name direction, an association can have more characteristics. With *multiplicity* one can state how many objects of a certain concept can be connected across an instance of an association. Multiplicity is visualized by using the following expressions: (1) for exactly one, (0..1) for one or zero, (0..*) for zero or more, (1..*) for one or more, or for example (5) for an exact number. In Figure 8-15 association with multiplicity is visualized.

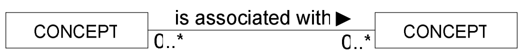


Figure 8-15: Multiplicity

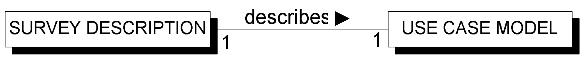


Figure 8-16: Example multiplicity

An example of multiplicity is represented in Figure 8-16. It is the same example as in Figure 8-14, only the multiplicity values are added. The figure reads as 'exactly one SURVEY DESCRIPTION describes exactly one USE CASE MODEL'. This implies that a SURVEY DESCRIPTION cannot describe zero or more than one USE CASE MODEL and a USE CASE MODEL cannot be described by zero or more than one SURVEY DESCRIPTIONS.

8.3.5. Aggregation

A special type of association is *aggregation*. Aggregation represents the relation between a concept (as a whole) containing other concepts (as parts). It can also be described as a 'has-a' relationship. In Figure 8-17 an aggregation relationship between OPEN CONCEPT and STANDARD CONCEPT is illustrated. An OPEN CONCEPT consists of one or more STANDARD CONCEPTS and a STANDARD CONCEPT is part of one OPEN CONCEPT.

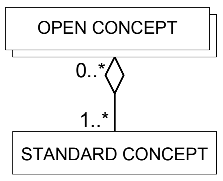


Figure 8-17: Aggregation

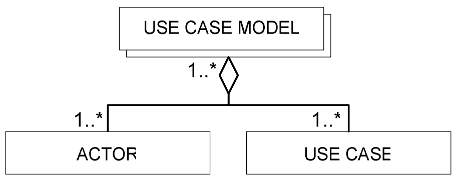


Figure 8-18: Example aggregation

In Figure 8-18 aggregation is exemplified by a fragment of the requirements capture workflow in UML-Based Web Engineering. A USE CASE MODEL consists of one or more ACTORS and USE CASES.

8.3.6. Properties

Sometimes the needs exist to assign properties to concepts. Properties are written in lower case, under the concept name, as is illustrated in Figure 8-19.

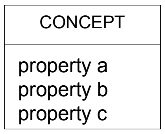


Figure 8-19: Properties

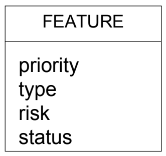


Figure 8-20: Example properties

In Figure 8-20 an example of a concept with properties is visualized. The concept FEATURE has four properties, respectively: priority, type, risk and status.

8.4. Process-data diagram

The integration of both types of diagrams is quite straightforward. Each action or activity results in a concept. They are connected with a dotted arrow to the produced artifacts, as is demonstrated in Figure 8-21. The concepts and activities are abstract in this picture.

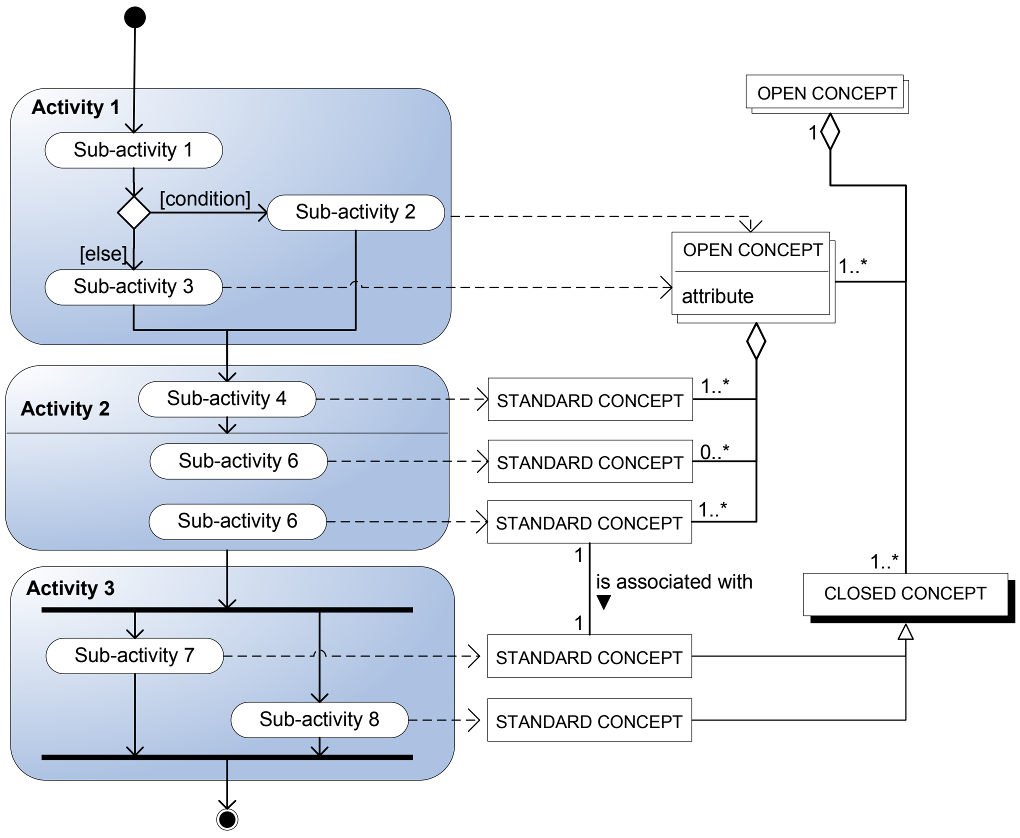


Figure 8-21: Process-data diagram

In Figure 8-22 an example of a process-data diagram is illustrated. It concerns an example from a complex project in the new developed method. Represented is the orientation phase.

Notable is the use of open and closed concepts. Since project management is actually not within the scope of this research, the concept CONTROL MANAGEMENT has not been expanded. However, in a complex project is RISK MANAGEMENT of great importance. Therefore, the choice is made to expand the RISK MANAGEMENT concept. For more information on the activities and concepts, see section 13.4.

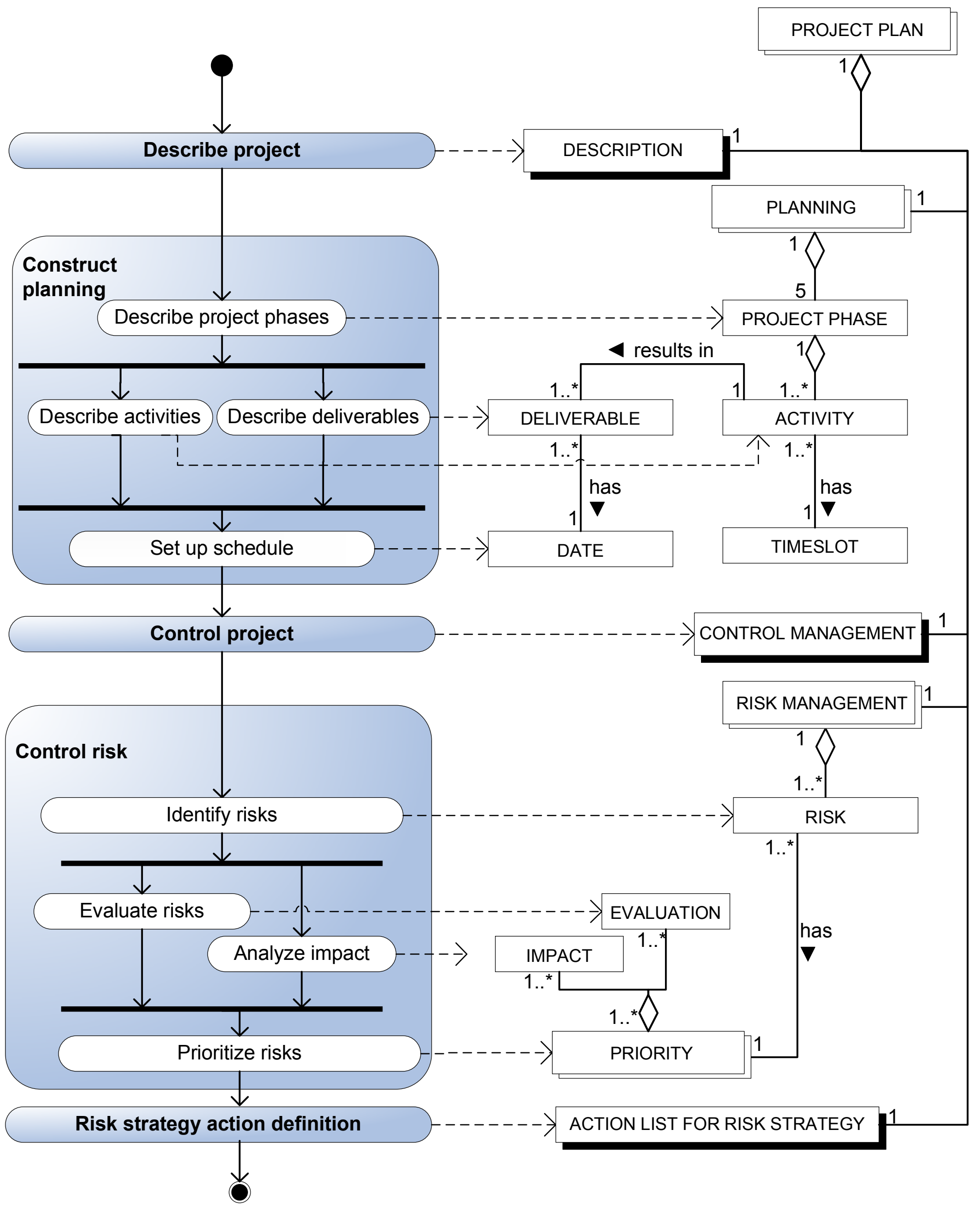


Figure 8-22: Example process-data diagram - Orientation phase in a complex project

Part IV: Method Base

In part IV, the method base is filled. This is done by analyzing the existing methods and developing process-data diagrams. The method fragments stored in the method base are used for developing the new method.

In chapter 11, the existing GX method is analyzed. Then, in chapter 12, the Unified Process is examined. Finally, in chapter 13, UML-based Web Engineering is studied.

9 Analysis of project phases at GX

9.1. Introduction

In this chapter, the first source which method fragments are stored in the method base is analyzed. The same analysis is performed on the Unified Software Development Process and UML-based Web Engineering. The way in which the method is modeled makes it possible to extract useful method fragments and use these in the 'method to-be'.

The methods used in the project phases can be clarified by analyzing the method rationale. Rossi, Tolvanen, Ramesh, Lyytinen and Kaipala (2000) distinguish two different levels of method rationale. On the level of method engineers there is the 'method construction rationale'. This method rationale is an explanation why certain types or constraints of the method are included in the constructed method. On the level of the method users, the method rationale explains why certain types or constraints of the method are or are not used in models. This method is called 'method use rationale'. Only the first method rationale is taken into account in this chapter. The latter is difficult to capture, since it has never been documented at GX. Moreover, it is difficult to elicit this knowledge, which can be months or years old, from peoples' minds. Because method construction rationale has never been collected systematically, this has to be done by means of semi-structured interviews and artifact analysis. Method rationale can be represented in different levels of formality (Rossi et al., 2000). In this chapter an informal approach is used due to the unstructured nature of the information.

The acquisition, orientation, definition and design phases of a WebManager implementation project are analyzed. Each of the project phases is described by using a process-data diagram. This meta-model describes the activities performed and artifacts delivered in each project phase. The design phase is further analyzed by outlining its method rationale.

9.2. Acquisition

9.2.1. Introduction

The acquisition starts with a lead or prospect. The account manager, who is the contact person of the customer, calls in the help of a pre-sales consultant to write the proposal. If necessary, the help of a software engineer is called in to give input on technical decisions. Usually two weeks are scheduled for the acquisition phase. It ends with the client's approval of the proposal. In Figure 9-1 the meta-model of the acquisition phase is illustrated. The activities and concepts are described in Table 9-1.

9.2.2. Meta-Model Proposal

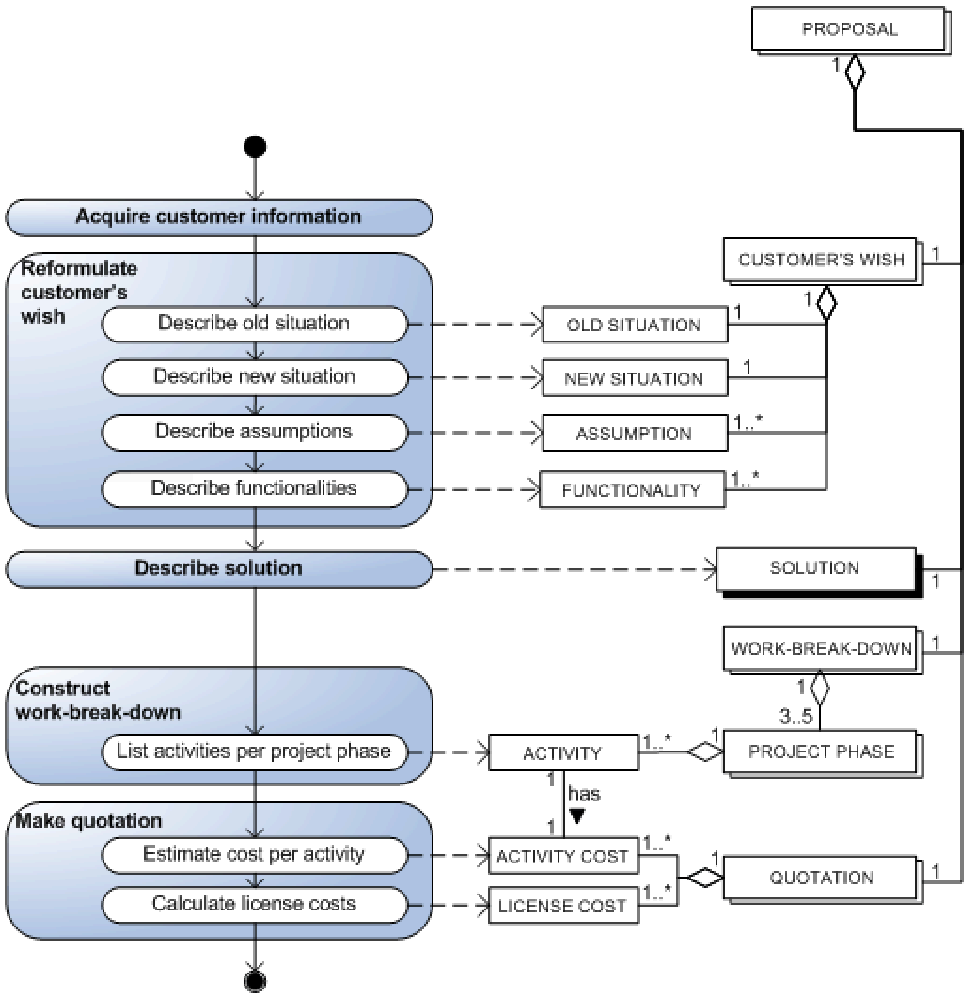


Figure 9-1: Meta-model acquisition phase

Table 9-1: Activities and sub-activities in the acquisition phase

Activity	Sub-Activity	Description
Acquire customer information		The pre-sales consultant has two ways of *acquiring customer information*. Usually the information is received indirectly via the account manager. However, with complex projects the pre-sales consultant will interview the customer himself.
Reformulate customer's wish	Describe old situation	In the OLD SITUATION the system as-is is described.
	Describe new situation	In the NEW SITUATION the system to-be is described.
	Describe assumptions	Describe ASSUMPTIONS to include some boundary conditions in the proposal.
	Describe functionalities	Describe the main FUNCTIONALITIES of the system.

Describe solution		The SOLUTION can be described in several ways. If concrete functionalities are described, these can be used as input. Per functionality a solution is provided, sometimes a standard WebManager solution, other times a customized solution. Often, the solution is entirely based on existing WebManager components. Standard WebManager component descriptions are included to show the customer how WebManager can provide several functionalities in the new system.
Construct WBD	List activities per project phase	SOLUTIONS are translated into ACTIVITIES and scheduled in de WORK-BREAK-DOWN per PROJECT PHASE.
Make quotation	Estimate cost per activity	Per ACTIVITY the ACTIVITY COST is estimated. This is part of the QUOTATION.
	Calculate license costs	The LICENSE COSTS are estimated. This is part of the QUOTATION.

9.3. Orientation Phase

9.3.1. Introduction

The orientation phase starts when the customer approves the proposal. At the end of the orientation phase, the project plan is delivered. The project manager writes this project plan. In Figure 9-2 the meta-model of the orientation phase is listed. This model is explained in Table 9-2.

9.3.2. Meta-Model Project Plan

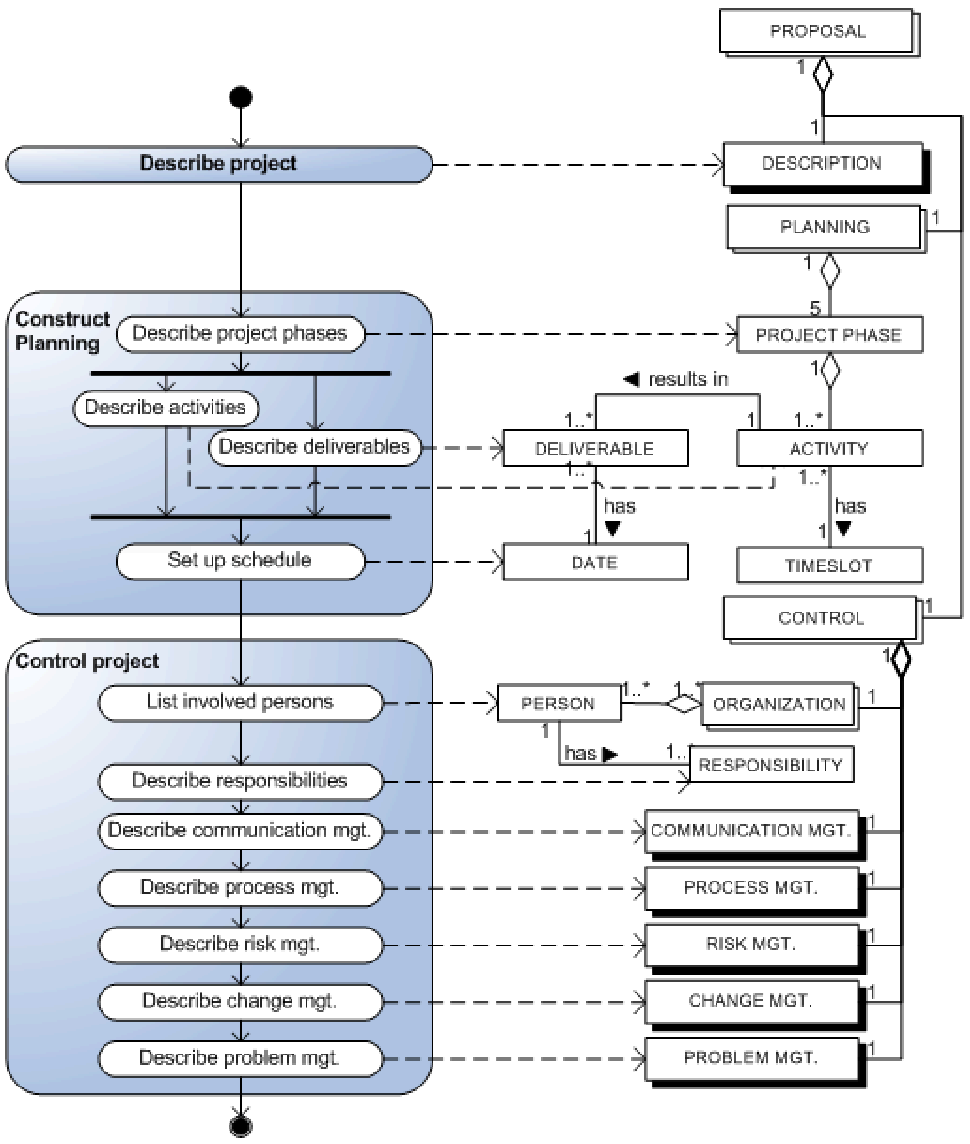


Figure 9-2: Meta-model orientation phase

Table 9-2: Activities and sub-activities in the orientation phase

Activity	Sub-Activity	Description
Describe project		Describing the project is done in terms of participants, targets, products, scope and assumptions. This information is derived from the proposal, but with more emphasis on the project management issues. The activity end in a project DESCRIPTION.
Construct planning	Describe project phases	The PLANNING is divided into five PROJECT PHASES, which should be shortly described.
	Describe activities	De ACTIVITIES are described and grouped into PROJECT PHASES
	Describe deliverables	The DELIVERABLES that result from the ACTIVITIES are described.
		For every DELIVERABLE a DATE is set and for each ACTIVITY a TIME SLOT is estimated.
Control project	List involved persons	All involved PERSONS are listed to describe the ORGANIZATION of the project.
	Describe responsibilities	RESPONSIBILITES per person are described here.
	Describe communication management	Issues like frequency of meetings are described in COMMUNICATION MANAGEMENT.
	Describe progress management	In PROGRESS MANAGEMENT is described how the progress is measured. This can be done by using the milestones as reference, or buy sending progress reports every week.
	Describe risk management	In RISK MANAGEMENT are settlements described concerning the risk handling. Mostly, this will be in the form of a risk log. Every project member should send identified risk to the project manager, who will add it to the risk log and decides what action to take.
	Describe change management	In CHANGE MANAGEMENT agreements about changing requirements, specifications, planning etc. are described.
	Describe problem management	In PROBLEM MANAGEMENT is described how mismatches between requirements and implementation are handled.

9.4. Definition Phase

9.4.1. Introduction

Requirements analysis is the process of finding out what has to be built. It is carried out in the specification phase of the GX design process. Requirements analysis is quite difficult, because users often do not know what they want. The task of the consultant is to translate the product vision on the new web application into a usable set of requirements.

The meta-model of the definition phase is illustrated in Figure 9-3 and described in Table 9-3.

9.4.2. Meta-Model

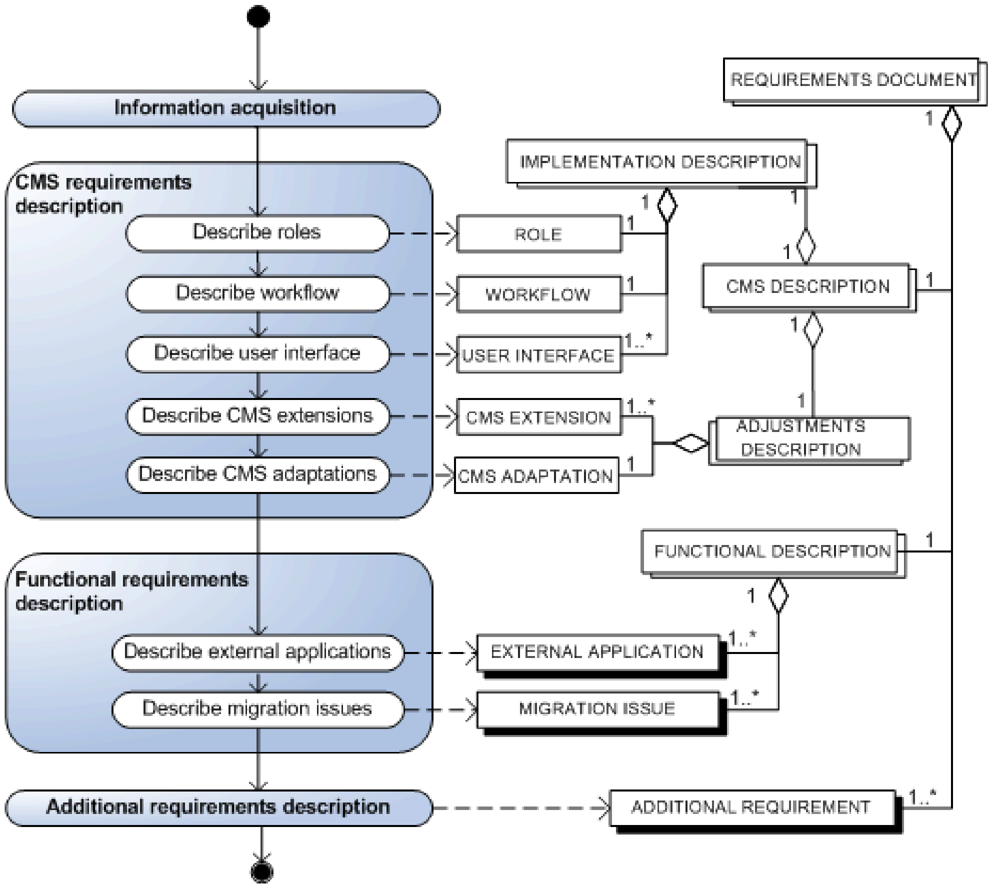


Figure 9-3: Meta-model definition phase

Table 9-3: Activities and sub-activities in the definition phase

Activity	Sub-Activity	Description
Acquire information		Information is acquired by using the proposal, communicating with customers and possibly organizing requirements sessions.
CMS requirements description	Describe roles	The ROLES that are used in the CMS are described here.
	Describe workflow	Describe the WORKFLOWS that have to be implemented in the system.
	Describe user interface	Describe USER INTERFACE in terms of design, guidelines etc.
	Describe CMS extensions	Custom CMS EXTENTIONS are described here.
	Describe CMS adaptations	Describe CMS customizations.
Construct WBD	Describe external applications	Interfaces with other systems are described in EXTERNAL APPLICATIONS.
	Describe migration issues	When content from the existing application has to bee migrated, this is described in MIGRATION ISSUES.
Additional requirements description		The non-functional requirements concerning, for example, security and maintenance, are described as ADDITIONAL REQUIREMENTS.

9.5. Design Phase

9.5.1. Introduction

In the design phase the architecture is defined and the core functionalities are designed. At the end of the design phase, the architecture document is delivered. This document is written by the software architect and is based on the requirements document. The requirements document describes *what* to be built, and the architecture document describes *how* this has to be done.

9.5.2. Method Rationale

The software architecture of a web application is specified by using the 4 + 1 View Model of Architecture. The five views of this model are illustrated in Figure 9-4: the logical view, the implementation view, the process view and the deployment view. In the meta-model of the design phase, the activities needed to describe the systems in terms of different, following the 4+1 View Model, are depicted.

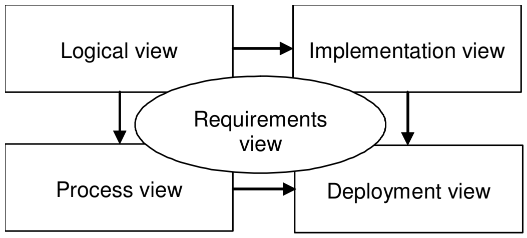


Figure 9-4: 4+1 View Model of Architecture, Adopted from the Software Architecture Document in RUP 5.5 (Rational Software Corp., 1999)

For the product WebManager a software architecture document already exists. It describes the architecture in terms of the five views. When the architecture of a web application has to be designed, the architecture of WebManager serves as the starting point. Several parts are rewritten (e.g. goal setting). Furthermore, in requirements section things like performance, security and maintenance issues are described. Based on this section the architect decides whether or not the architecture has to be changed. If so, components and processes can be added, changed or removed. If not, the rest of the architecture doesn’t need to be changed anymore.

9.5.3. Meta-model

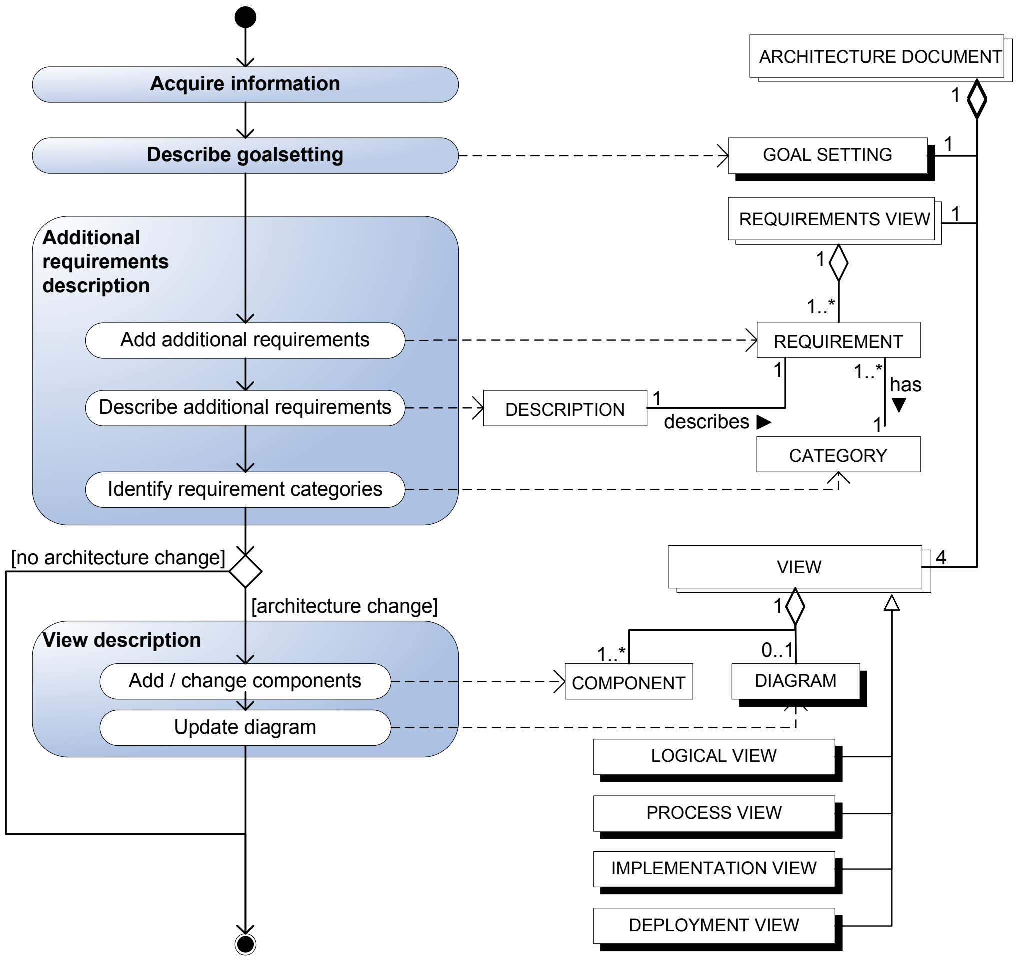


Figure 9-5: Meta-model design phase

Table 9-4: Activities and sub-activities in the design phase

Activity	Sub-Activity	Description
Information acquisition		Information acquisition is done by studying the requirements document and by meetings with the consultants and project managers.
Describe goalsetting		The goals, scope and stakeholders are described in the GOALSETTING.
Additional requirements description	Add additional requirements	If necessary, add additional non-functional REQUIREMENTS to the architecture.
	Describe additional requirements	Describe the additional REQUIREMENTS
	Identify requirement categories	Provide the REQUIREMENTS with a CATEGORY. The CATEGORIES used are security, scalability & performance, availability, maintainability, tractability and usability.

Branch	*Architecture change*	*When all additional requirements are described, the decision has to be taken whether or not the software architecture had to be adapted to the new web application*
View description	Add/change components	The architect can add, change, or remove COMPONENTS from the existing software architecture. COMPONENTS belong to a view, see [1]. The word component is used in the diagram, but in case of the process VIEW, it should be read as "process".
	Update diagram	Sometimes a view is illustrated by means of a DIAGRAM. When components are changed, this DIAGRAM should also be updated.

The four views are the logical view, process view, implementation view and deployment view. The *logical view* describes the structure of the web application in terms of functional components. These components often are divided into different layers. The *process view* describes the web application in terms of processes and the communication between these processes. Important in this setting are performance, scalability and throughput capacity. In the *implementation view* the architecture is presented from the perspective of software development. The software is divided into components which can be separately developed, tested and possibly reused. The *deployment view* is concerned with the physical distribution of processes over a set of processors within a certain network topology. This view usually depends heavily on the information technology infrastructure at the customer.

10 Unified software development process

10.1. Introduction

The second source form which fragments are stored in the method base, is the Unifies Software Development Process (Unified Process). The Unified Process is “a generic process framework that can be specialized for a very large class of software systems, for different application areas, different types of organizations, different competence levels, and different project sizes” (Jacobson, Booch & Rumbaugh, 1999). A web application has a relatively small development process. Therefore, the Unified Process should be tailored into a situational method.

Firstly, a brief history overview of the Unified Process is given. Secondly, the main distinguishing aspects of the Unified Process are described. Then, the lifecycle of the entire process is outlined. Finally, the requirements analysis workflow and analysis workflow are described by means of a meta-model.

10.2. History

The Unified Process originates from the sixties. In 1967 Ericsson started with modeling systems as a set of interconnected blocks, which corresponded directly to a simplified version of UML class diagrams (Booch, Jacobson & Rumbaugh, 1999). In 1987 Objectory AB was established in Stockholm by Ivar Jacobson, a former employee of Ericsson. Although the concept of use cases was raised by Ericsson, Objectory was the first one who used the diagramming technique of use case modeling. When Objectory AB was acquired by the Rational Software Corporation, Jacobson could bundle his knowledge with Grady Booch, the author of the Booch method, and James Rumbaugh, the developer of the Object Modeling Technique. Together, they developed the Unified Modeling Language, which in 1997 was recognized as standard by the Object Management Group. UML was used as modeling technique in the Rational Objectory Process, which was developed by Rational during this period. In 1998 the newest version of the Rational Unified Process was published. Also, the name was changed into the Unified Software Development Process, in order to reflect the unification of development approaches and methods (Booch et al., 1999).

10.3. Distinguishing aspects

10.3.1. Introduction

The authors find three aspect of the Unified Process are highly distinguishing. These unique aspects are:

- use-case driven;
- architecture-centric;
- iterative and incremental.

These three concepts are all equally important. According to Jacobson, Booch and Rumbaugh (1999), architecture provides the structure in which to guide the work in the iterations, whereas use cases define the goals and drive the work of each iteration. In the following of this paragraph the three aspect are further expounded.

10.3.2. Use-case driven

In the Unified Process use cases are used to capture the functional requirements of a system. Use cases are expressed linguistic, in use case descriptions, or graphically, in a use-case model.

A use-case model consists of several use cases and captures the entire functionality of the system to be built. Use cases do not only describe what the system is supposed to do, but also describes what it has to do for each user. In doing this, it drives the requirements specification, design, implementation and testing of a system, that is, use cases are specified, use cases are designed, and at the end use cases are the source from which the testers construct the test cases (Jacobson, Booch & Rumbaugh, 1999).

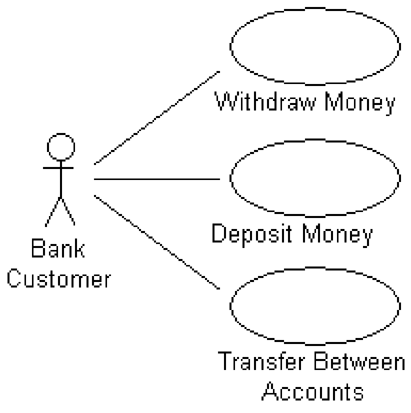


Figure 10-1: Use-Case Model

In Figure 10-1 a simple example of a use-case model is illustrated. It consists of one actor, the *bank customer*, and three use cases: *withdraw money*, *deposit money* and *transfer between accounts*. The actor is connected to the use cases with associations. A use-case model contains one or more actors. Actors and use cases can be modeled by using generalization in the same way as is used in a class diagram. Also, two types of relations between use cases exist, namely *extend* and *include*. An extend relationship is used to model a part of the system that is conceived by the user as optional system behavior. An include relationship, on the other hand, means that the behavior of the included use case is explicitly used by the base use case.

10.3.3. Architecture-centric

The software architecture describes the form in which the functions of a system, as described in the use cases, have to be developed. The architecture and use-cases must evolve in parallel, because they both influence each other. According to Jacobson, Booch and Rumbaugh (1999) a software architecture is developed as follows:

- The architect makes an outline of the system, starting with the use-case independent parts. However, a global understanding of the use cases is required.
- The software architect specifies the use cases that identify the main functions of a system. Based on the specified use cases, subsystems, classes and components are developed.
- The specification of use cases leads to a better understanding of how the architecture should be modeled. On the other hand, developing the software architecture leads to changes and refinements of the use cases.

10.3.4. Iterative and Incremental

The Unified Process is divided into several mini-projects. Mini-projects are described as iterations that result in an increment. Iterations and increments differ from each other in that iterations refer to the planned and controlled steps that are taken in the workflows, as increments refer to the growth of the system to be developed. Setting up the iterations depends on which groups of use cases should be developed and the risks that are concerned with this. The process of working in an iteration is as follows:

1. Identify and specify the use cases that need to be implemented.
2. Create a design by using the specified architecture as a guide.
3. Implement the design in several components.
4. Verify that the components actually do what is described in the use cases.

When the iteration is finished successfully, a new iteration can be started. If it is not finished successfully, the iteration has to be revisited and redone with a new approach.

10.4. Lifecycle

The Unified Process can be divided into four phases and five workflows. Each phase is subdivided into two or more iterations. After each iteration a release is delivered. This release can consist of a working part of the system, but it can also be the use-case model or architecture artifacts. In Figure 10-2 the lifecycle of the Unified Process is illustrated.

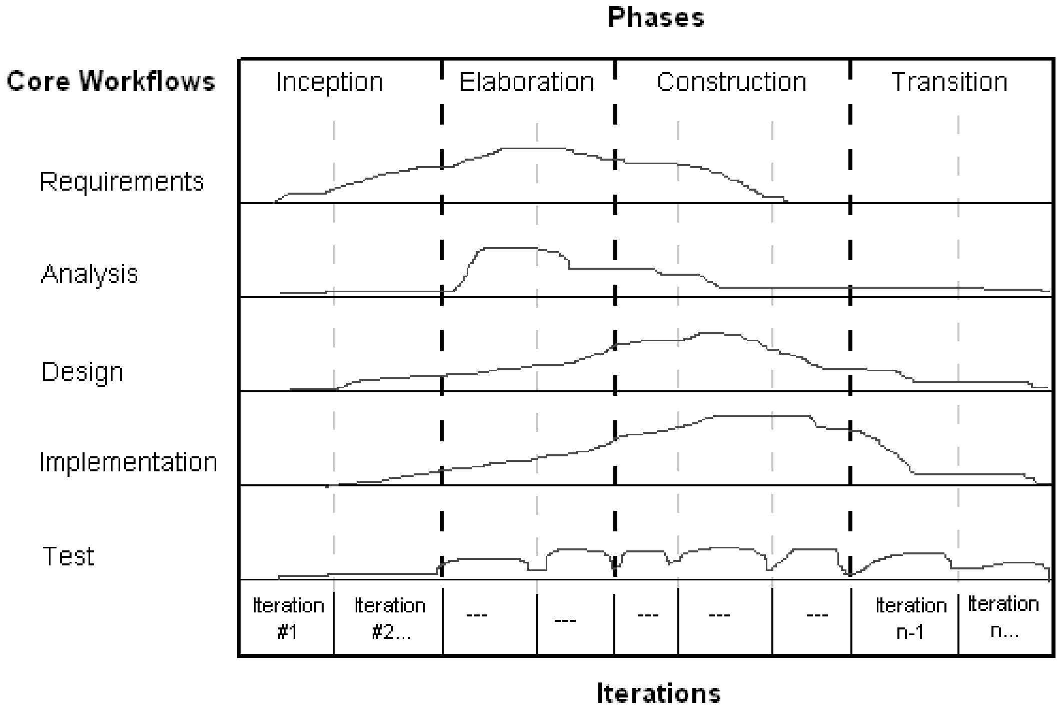


Figure 10-2: Unified Process lifecycle (Jacobson, Booch & Rumbaugh, 1999)

Four phases exist: inception, elaboration, construction and transition. Each phase ends in a milestone, where an earlier defined set of artifacts should be delivered. The five workflows are requirements, analysis, design, implementation and test. For this research project the requirements and analysis workflows are important, especially in the inception and elaboration phases.

10.5. Requirements workflow

10.5.1. Introduction

The requirements workflow starts with listing the candidate requirements and ends with a structured use case model. People active in this phase are system analysts, architects, use-case specifiers and user-interface designers. In this section the method rationale and process-data model of the requirements workflow are provided.

10.5.2. Method rationale

Requirements capture is difficult, caused by communication problems between customer and developer. Customers find it hard to express their wishes regarding the new system in terms of clear requirements. Developers have difficulties expressing themselves in a language the customer understands. To overcome this problem a systematic process has been developed to capture the system's requirements. This systematic process can handle different starting points, ranging from a vague vision to a detailed list with requirements. Although the approach has to be adapted to the starting point, several steps in the workflow should be followed:

- List candidate requirements
  The resulting artifact of this step is a list with features. These features are actually ideas that possibly become real requirements. Except a description, a feature can consist of things like status, cost, priority and risk.
- Understand system context
  Two ways of expressing the context of a system are described: domain modeling and business modeling. A domain model describes the context in the form of objects and the relations between them. A business model is larger than a domain model. It describes not only the objects, but also the business processes of the system context. This step ends in either a domain model or a business model.
- Capture supplementary requirements
  Non-functional requirements like security and maintainability should be connected to the use case they are relevant for. Non-functional requirements that can't be captured in use cases should be managed separately in a list of supplementary requirements.
- Capture functional requirements
  The functional requirements are captured in use cases, as is described earlier. A use-case model is the resulting artifact of this step.

Finally, the described steps are input for the prototype. This prototype can be used for verification purposes. Reviewers verify that each user interface (a) allows the actor to navigate properly, (b) provides a consistent look and feel and a consistent way of working with the user interface, and (c) complies with relevant standards such as colors, size of buttons, and placement of toolbars (Jacobson, Booch & Rumbaugh, 1999).

10.5.3. Meta-Model

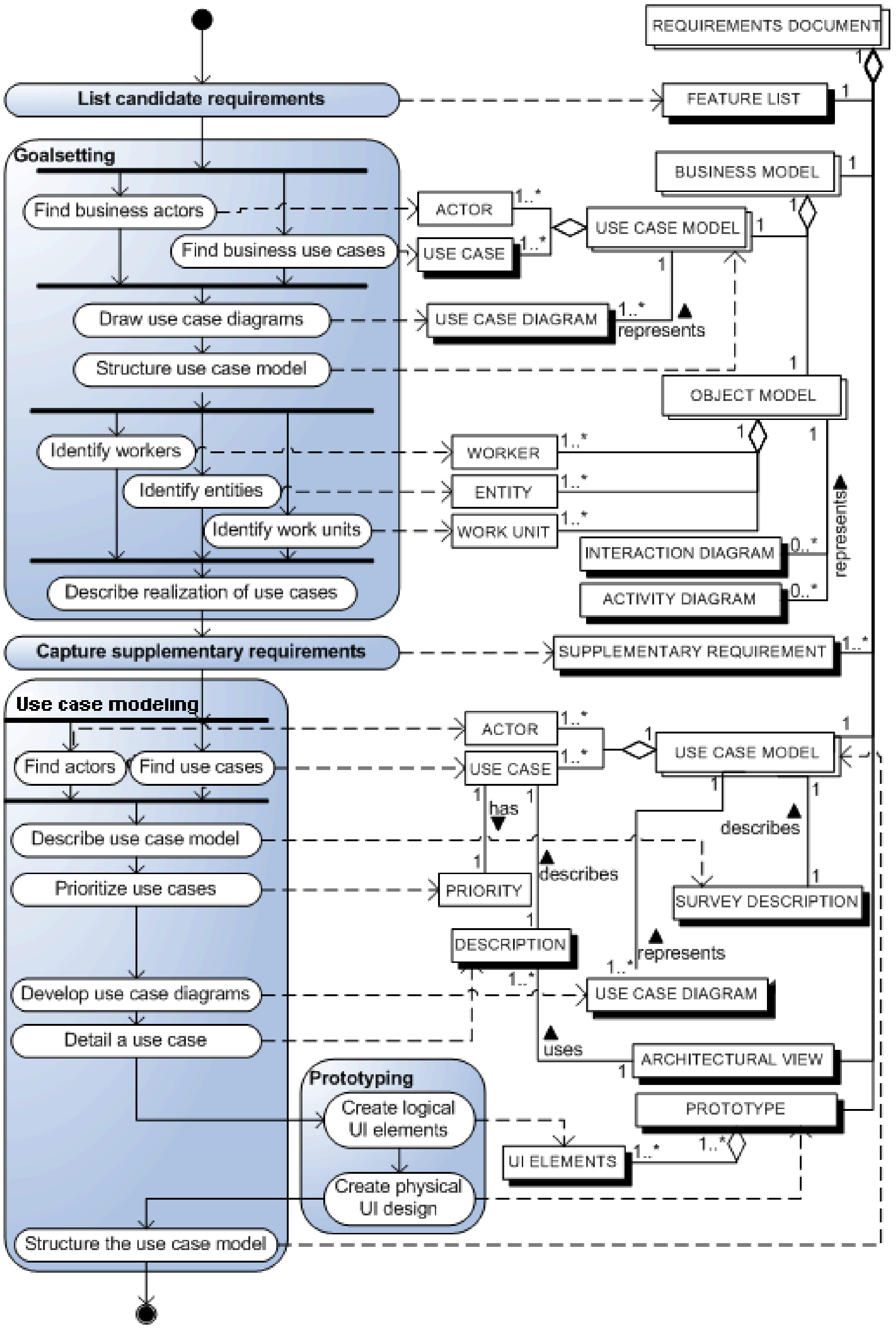


Figure 10-3: Meta-Model Requirements Analysis Unified Process

Table 10-1: Activities and sub-activities in the UP requirements workflow

Activity	Sub-Activity	Description
List candidate requirements		The requirements modeling process starts with *listing the candidate requirements* in the FEATURE LIST. This FEATURE LIST has one or more features, which describes the candidate requirement in terms of status, cost, priority and risk.
Understanding the business context	Find business actors	Describe BUSINESS ACTORS to represent business customers.
	Find business use cases	Describe BUSINESS USE CASES to represent business processes.
	Draw use case diagram	Represent the USE CASE MODEL by drawing a USE CASE DIAGRAM.
	Structure use case model	Structure the USE CASE MODEL by evaluating use case descriptions and restructuring the use case model by means of *extend* or *include* relationships.
	Identify workers	Describe business WORKERS, who are realizing the business use case.
	Identify entities	Describe business ENTITIES. Business ENTITIES are things, for example an invoice, which can be manipulated (created, accesses, changed, deleted etc.)
	Identify work units	Describe business WORK UNITS. A WORK UNIT is a set of ENTITIES.
	Describe realization of use cases	Describe realization of use cases in INTERACTION or ACTIVITY DIAGRAMS. By developing these diagrams, the USE CASES are realized by the WORKERS, ENTITIES and WORK UNITS. These diagrams are not further detailed in this meta-model.
Capture supplementary requirements		All non-functional requirements that can be related to one of the use cases should be listed in the SUPPLEMENTARY REQUIREMENTS. This concept class is not expanded, because several alternatives exist to model this. Jacobson, Booch and Rumbaugh (1999) make the following categorization: ▪ Interface requirements ▪ Physical requirements ▪ Design constraints ▪ Implementation constraints Per category several requirements can be listed, expressed in natural language. Although the capturing of supplementary requirements is modeled before the use-case modeling, it is likely that this list will grow during the other main activities.
Use-case modeling	Find actors	Derive ACTORS from the business ACTORS in the BUSINESS MODEL.
	Find use cases	USE CASES are found by identifying every role of each WORKER who participates in a business use-case realization and who uses the information system.
	Develop use case model	Use ACTORS and USE CASES to develop the USE CASE MODEL

	Describe use-case model	Describe the USE CASE MODEL in a SURVEY DESCRIPTION.
	Prioritize use cases	Give USE CASES a PRIORITY to determine which are most important and need to be developed early.
	Develop use case diagrams	Draw USE CASE DIAGRAMS.
	Detail a use case	Provide the USE CASES with a USE CASE DESCRIPTION. For more information on USE CASE DESCRIPTIONS, see Booch, Jacobson and Rumbaugh (1999).
	Structure the use case model	Structure the USE CASE MODEL by evaluating use case descriptions and restructuring the use case model by means of *extend* or *include* relationships.
Prototyping	Create logical UI elements	Create the UI ELEMENTS that are manipulated by the actors when performing a use case.
	Create physical UI design	Build a PROTOTYPE using, for example, a rapid prototyping tool. The most important use cases concerning usability should be translated to an executable graphical user interface. For the rest, paper sketches can be used. According to Booch et al. (1999), the reviewer should use the PROTOTYPE S to verify that each user interface: ▪ allows the actor to navigate properly; ▪ provides a consistent look and feel, and a consistent way of working with the user interface, such as tab ordering and accelerator keys; and ▪ complies with relevant standards such as colors, size of buttons, and placement of toolbars.

10.6. Analysis Workflow

10.6.1. Introduction

In the analysis workflow the requirements described in the requirements analysis are refined and structured. This is done by the architect, use-case engineer and component engineer. The workflow results in an analysis model. This is a conceptual object model including analysis packages, analysis classes, use case realizations and the architectural view of the analysis model.

10.6.2. Method Rationale

During the analysis workflow the captured requirements are specified, formalized and structured. This is done by analyzing the architecture, the use cases, the classes and the packages. The result can be seen as an early version of the design model.

Several high-level activities can be identified in the analysis workflow:

- Architectural analysis
  Architectural analysis has as purpose to outline the analysis model and the architecture by identifying analysis packages, obvious analysis classes, and common special requirements.
- Use case analysis

Each use case needs to be realized terms of the participating analysis classes by stating the behavioral elements of each class.

- Class analysis
  The behavioral requirements stated in the preceding activity are specified and integrated into each class by creating consistent responsibilities, attributes, and relationships for each class.
- Package analysis
  Package analysis aims at (a) making the analysis packages as independent of other packages as possible, (b) fulfilling its purpose of realizing some use cases, and (c) describing dependencies so that the effect of future changes can be estimated (Jacobson, Booch & Rumbaugh, 1999).

10.6.3. Meta-Model

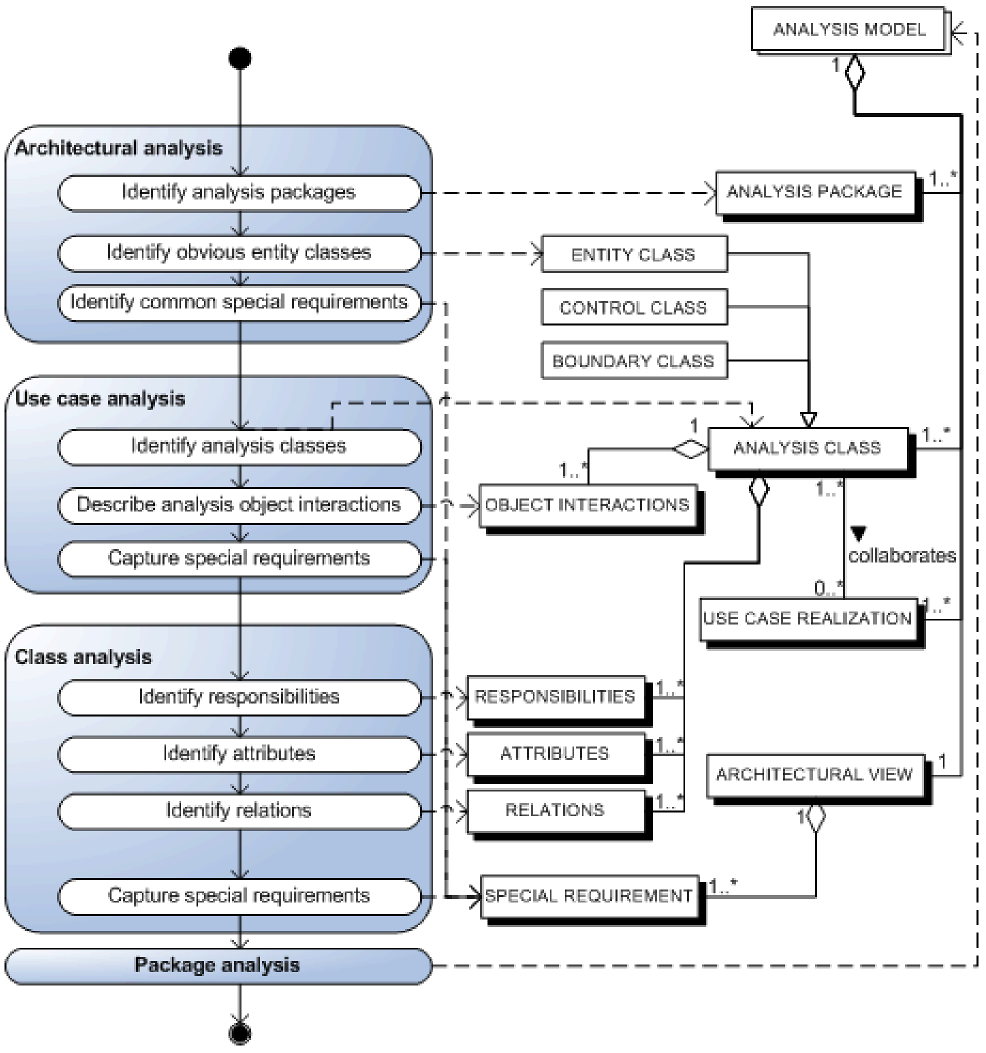


Figure 10-4: Analysis Workflow

Table 10-2: Activities and sub-activities in the UP analysis workflow

Activity	Sub-Activity	Description
Architectural analysis	Identify analysis packages	Identify ANALYSIS PACKAGES to organize the analysis model in manageable packages. The identification of packages is done by selecting use cases that (a) support a specific business process, (b) support a specific actor, or (c) are related by generalizations end extend-relationships.
	Identify obvious entity classes	Identify obvious ENTITY CLASSES in a preliminary proposal based on the information obtained during the requirements capture. Later on these entity classes will be expanded and more detailed.
	Identify common special requirements	Identify SPECIAL REQUIREMENTS like, for example, constraints on persistence, security features and fault tolerance. The key characteristics of every special requirement should be described to assure that it is handled appropriately in the design and implementation phases.
Use Case Analysis	Identify analysis classes	Identify ANALYSIS CLASSES. Three ANALYSIS CLASSES exist, which are CONTROL, ENTITY and BOUNDARY CLASSES. Control CLASSES are used for coordination, sequencing, transactions, and control of other objects. ENTITY CLASSES are to model information that is long-lived and persistent. BOUNDARY CLASSES are used to model the interaction between users and system. ANALYSIS CLASSES that collaborate together in a use-case realization should be modeled in one class diagram.
	Describe analysis object interactions	Describe OBJECT INTERACTIONS in collaboration diagrams.
	Capture special requirements	The SPECIAL REQUIREMENTS that are identified during the use case analysis, but need to be handled in a later phase are captured here.
Class analysis	Identify responsibilities	Describe the RESPONSIBILITES of each class, based on its role in the use-case realizations. This is done by analyzing the class and interaction diagrams of these classes.
	Identify attributes	Identify the ATTRIBUTES of each class.
	Identify relations	Identify RELATIONS between classes. RELATIONS are associations, aggregations and generalizations
	Capture special requirements	The requirements that come up in this activity but can't be handled in the analysis workflow are added to the SPECIAL REQUIREMENTS.
Package analysis		Although packages have been identified in the first main activity, constantly new ANALYSIS PACKAGES, CLASSES and COMMON REQUIREMENTS are found during the analysis. Main issues are managing dependencies between packages and maintaining the cohesiveness within packages.

11 UML-Based Web Engineering

11.1. Introduction

UML-Based Web Engineering method (UWE) is the third and last source for the method base. UWE is a systematic, prescriptive, user-centric, UML-based, iterative and incremental method for adaptive hypermedia systems (Koch, 2001). Brusilovsky, as cited in Koch (2001), defines adaptive hypermedia systems as: "adaptive hypermedia systems are hypermedia systems which reflect some features of the user in a user model and use this model by adapting various visible aspects of the system to the user". Koch (2001) describes web applications a subset of adaptive hypermedia systems. For the purpose of consistency, the term *web applications* is used in the rest of this chapter.

The outline of this chapter is as follows. Firstly, an overview of the method's main characteristics is given. Than, the method's lifecycle is outlined. Finally, activity-class diagrams of the requirements capture, risk management, iteration planning and validation workflows are given.

11.2. Overview of UWE

11.2.1. UWE and the Unified Process

UWE is based on the Unified Process (Jacobson, Booch & Rumbaugh, 1999). Differences are:

- the specialization of the Unified Process for the development of web applications;
- the extension of the development cycle with a maintenance phase;
- the addition of two supporting workflows, project management and quality management;
- extending quality control management with requirements validation and design verification in addition to only testing;
- proposing a stereo-type based extension UML for web applications; and
- including a systematic method for the analysis of web applications.

11.2.2. The software development process of UWE

The software development process of UWE consists of five phases, as is shown in Figure 11-1. The first four phases, inception, elaboration, construction and transition are the same as the phases described in the Unified Process. The fifth phase is a maintenance phase. This phase begins when the first version of the web application is delivered and ends when it is not used anymore. Adjustments, improvements and updates of the system are handled in the maintenance phase. Every phase ends with a milestone, which are respectively: life cycle objectives, life cycle architecture, initial operational capability, product release, and product cessation. These milestones consist on the one hand of a pre-defined set of artifacts, such as models, code or documents; and on the other hand of the decisions that are taken before the next phase can start.

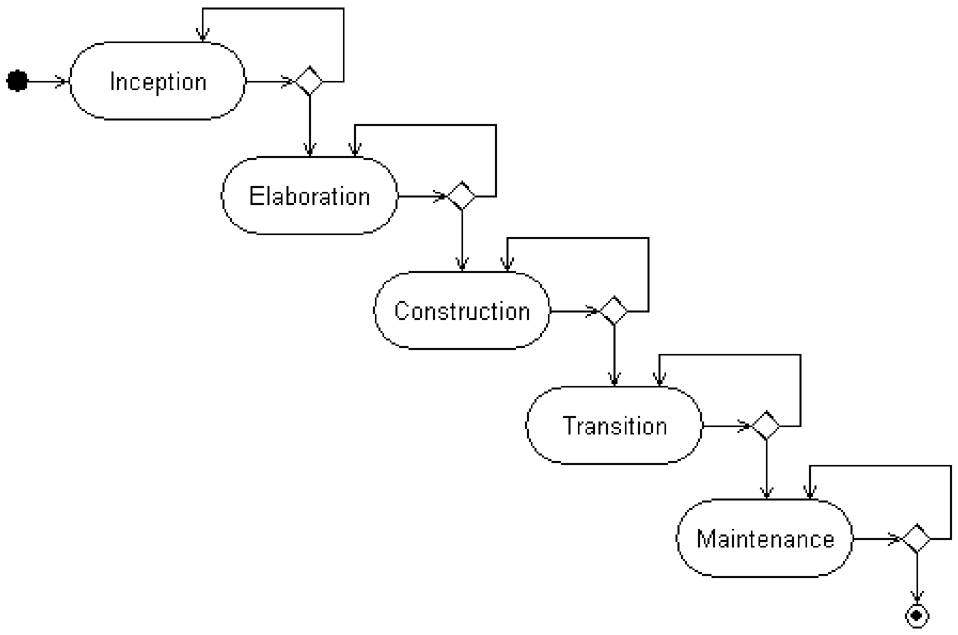


Figure 11-1: UWE process (Koch, 2001)

Every phase follows the *iteration workflow* which is described in the next section. In this research project the focus will be on the inception phase. Therefore, this phase will be described in detail later on in this chapter.

11.2.3. Iteration workflow

UWE is an iterative process. The amount of iterations performed in every phase is variable. Each iteration consists of a set of process workflows, which is called the *iteration workflow*. In Figure 11-2 this iteration workflow is illustrated. Three workflow groups are represented: the development process, which is the main workflow group; and project management and quality management, the supporting workflow groups. The development process consists of three workflows, namely requirements capture, analysis and design, and implementation. Project management embodies risk management, iteration planning and iteration evaluation. Finally, quality management consists of the validation, verification and testing workflows. The flow of control is depicted with continued lines and dependencies between the workflows are depicted with dashed lines.

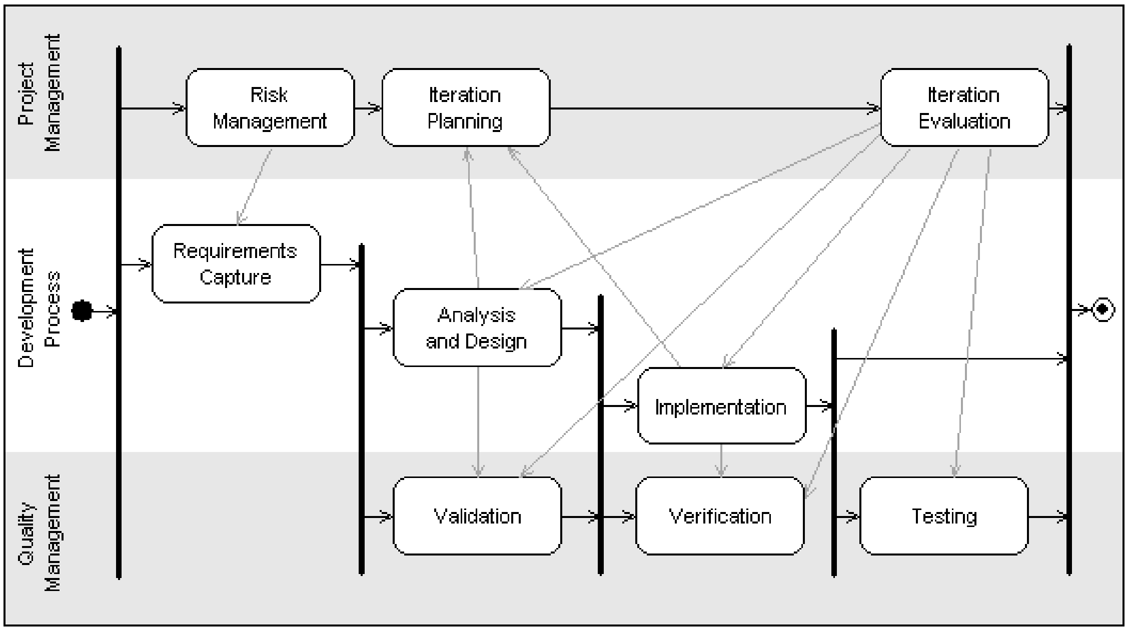


Figure 11-2: Iteration workflow (Koch, 2001)

11.2.4. Inception phase

The first phase of the UWE process has as main objective to establish the feasibility of a project. The project starts with an idea, which has to be developed and evaluated. Goals of a feasibility study are: (a) defining the most important functional requirements; (b) outlining the budget and costs; (c) producing a draft schedule plan; and (d) identifying the main non-functional requirements of the web application. Sometimes also a prototype is being developed, but this is generally only the case when it concerns a large project.

The workflows that are focused on in the inception phase are requirements capture, analysis and design, risk management, project planning and validation of the requirements. The milestone of this phase, the life cycle objectives, consists of the following deliverables:

- first version of the problem domain model;
- a first version of the use case model;
- a first draft of the architecture description;
- a prototype to prove the concepts or a new technology (optional);
- a risk study;
- a plan for the whole project;
- a business case, including success criteria, risk analysis, and budget estimation; and
- an architecture validation and a requirements review report (Koch, 2001).

Important factors that need to be taken into account are the current information, information sources and information structure; current applications; stakeholders; resources; technological limitations; and constraints.

In the following sections four workflows are further analyzed, namely requirements capture, analysis and design, risk management, iteration planning and validation. The original metamodels are in the form of activity diagrams. These diagrams are translated into activity-class diagrams in order to be consistent with the diagrams developed in the other chapters.

11.3. Requirements capture workflow

11.3.1. Introduction

In the requirements capture workflow four workers are active: business experts, architects, hypermedia analysts and user interface designers. The workflow starts with identifying the users and delivers in the end a structured use case model and a user interface prototype.

Two types of requirements exist: functional requirements and non-functional requirements. The functional requirements are categorized by Koch (2001) into the following types:

- requirements related to content;
- requirements related to structure;
- requirements related to presentation;
- requirements related to adaptation; and
- requirements related to the user.

Three activities can be recognized in the requirements capturing workflow to capture and translate the requirements described above. The first activity is *user and domain modeling*. Actions in this activity include identifying users (requirements related to the user), eliciting information needs (requirements related to content) and capturing a common vocabulary.

The second activity is about what kind of application needs to be developed, referred to as *application modeling*. The remaining requirement types are elicited and described here. The activity consists of eliciting navigational needs (requirements related to structure), eliciting user interface needs (requirements related to presentation), eliciting adaptation capabilities (requirements related to adaptation), eliciting additional requirements and prototyping the user interface.

Finally, the last activity, *use case modeling*, consists of modeling the captured requirements into use cases. This activity consists of finding actors and use cases, detailing, prioritizing and structuring the uses cases. Use cases drive the development process, in the sense that they are used to model requirements, are specified during analysis and design, and used as source during testing.

11.3.2. Meta-model

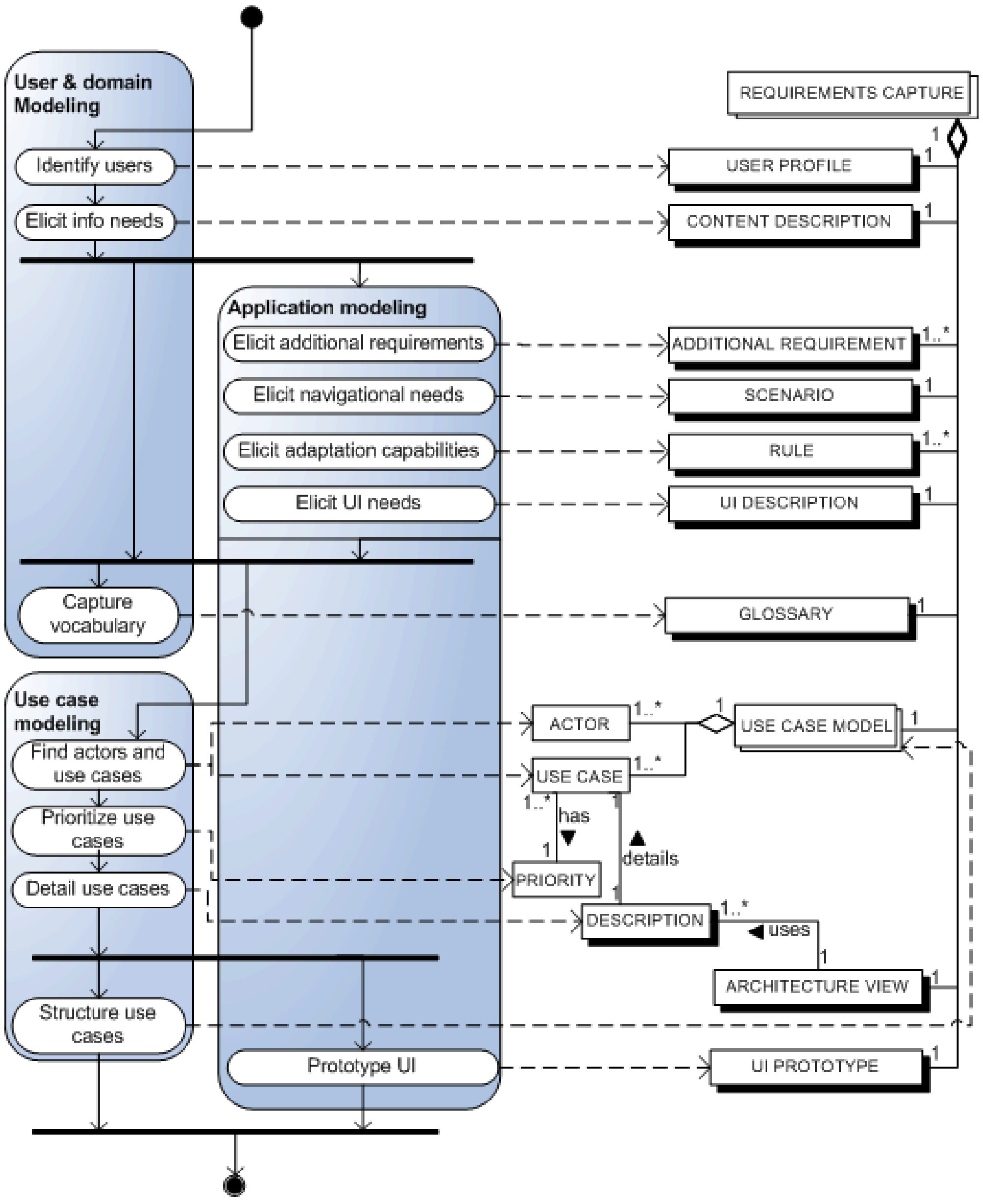


Figure 11-3: Meta-model requirements capture workflow

Table 11-1: Activities and sub-activities in the UWE requirements workflow

Activity	Sub-Activity	Description
User and domain modeling	Identify users	Identify USERS to reveal user's tasks, preferences, interests and knowledge of the domain. Retrieving this information can be done by, for example, interviewing, artifact analysis or brainstorm sessions. The result is a USER PROFILE.
	Elicit information needs	Find out the user's information needs resulting in a CONTENT DESCRIPTION.
	Capture vocabulary	Describe a common GLOSSARY that can be used by all stakeholders.
Capture supplementary requirements	Elicit additional requirements	Describe ADDITIONAL REQUIREMENTS. Koch (2001) identifies the following types: budget constraints, time constraints, hardware constraints, software constraints, design constraints, user modeling constraints, implementation constraints, performance, security, availability, ergonomics and usability.
	Elicit navigational needs	Elicit navigational needs to find out how information should be accessed in the web application. This can be retrieved by interviewing, exploring best practices, following guidelines, etc. The result of this action is a set of SCNARIOS, which consist of a description of the typical navigational behavior of the user.
	Elicit adaptation capabilities	Identify the required adaptive capabilities that are needed. A technique to find this out is observing users while they are interacting with a similar non-adaptive system. This results in a set of ADAPTATION RULES. In the first iteration this is in natural language. Later on, this can be formalized.
	Elicit UI needs	Find out to find out how information and navigation assistance should be presented to the user. Interviewing the customers or studying existing web applications can be done to retrieve these needs. The result is a USER INTERFACE DESCRIPTION, which can later on be extended with a user interface prototype.
	Prototype UI	During the prototyping, a first approach of the interface is developed, based on the results from the activities application modeling, but also on the results of the use case modeling. Therefore, developing a USER INTERFACE PROTOTYPE is carried out in the end.
Use-case modeling	Find actors and use cases	Find ACTORS and USE CASES by scheduling workshops and interviews.
	Prioritize use cases	Here is determined what PRIORITY each USE CASE will have in the development. The result of this can be seen in the ARCHITECTURE VIEW, which contains only the important USE CASES.
	Detail a use case	Provide the USE CASES with a DESCRIPTION to describe the flow of events. This action results in a detailed description consisting of text and use-case diagrams.
	Structure use cases	Establish relationships between USE CASES and ACTORS, and generalizations of between ACTORS to structure the USE CASES in the USE CASE MODEL.

11.4. Analysis and design workflow

11.4.1. Introduction

In the analysis and design workflow, the requirements description produced in the preceding workflow has to be translated into a specification that describes how to implement the web application. This is done by the business expert, architect, hypermedia analyst, and user interface designer. Artifacts delivered in this workflow are the design view of the architecture, the conceptual model, the user model, the navigation model, the presentation model, the adaptation model, design classes, subsystems and interfaces.

This analysis and design workflow consists of two parts. The analysis part focuses on functional requirements. The other requirements are handled in the design part. During this activity, the analysis results are adapted to the conditions described in the non-functional requirements.

11.4.2. Meta-model

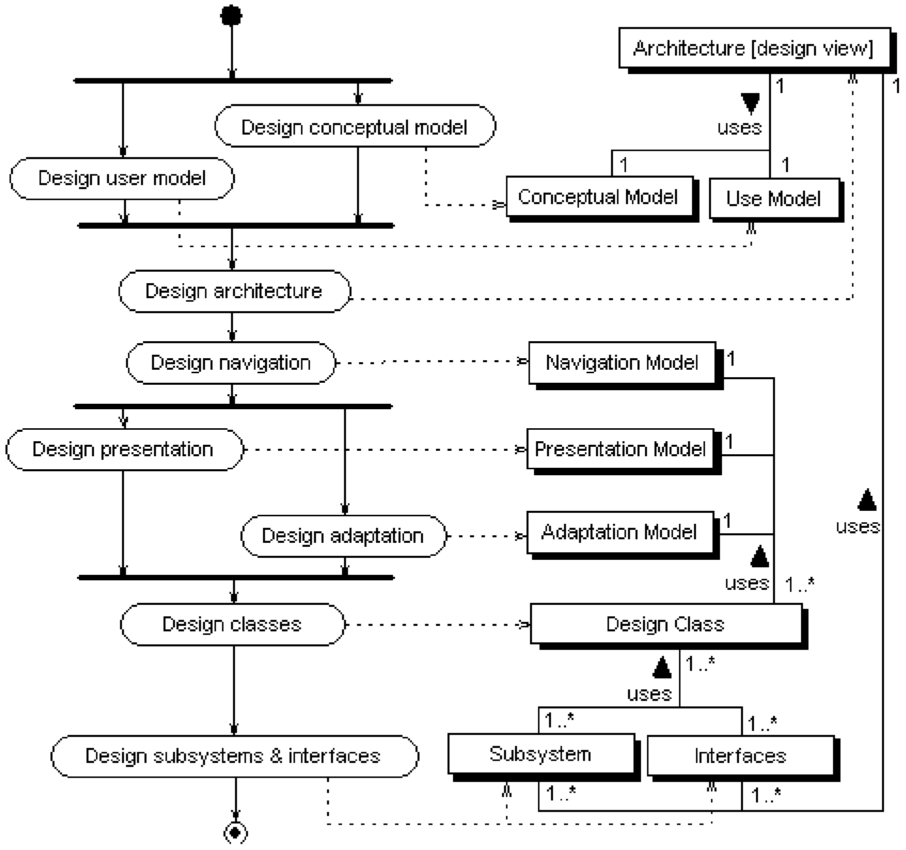


Figure 11-4: Meta-model analysis and design workflow

Table 11-2: Activities in the UWE analysis workflow

Activity	Description
Design conceptual model	During the conceptual design a business model is being built. All concepts relevant to users and user groups identified in the requirements workflow are included here. Navigation, presentation and interaction aspects are covered later. The CONCEPTUAL MODEL is visualized in a UML class diagram.
Design user model	Build a USER MODEL that represents knowledge, goals and/or individual features, such as preferences, interests and tasks of the users". The user model supports the adaptation of the web application in the sense that the application dynamically adjusts itself to the user. A user model is represented with a UML class diagram, where the classes describe the user attributes and their relationship to the conceptual model.
Design architecture	Designing architecture aims at describing the design view of the architecture by identifying: ▪ subsystems and their interfaces; ▪ design classes, that are relevant for the architecture; ▪ generic mechanisms to handle functional and non-functional requirements; and ▪ reuse possibilities, such as reusing parts of similar systems or general software products (Koch, 2001). The architecture can be represented as a simple drawing of the different parts of the system. These parts are elaborated on later in the process.
Design navigation	During the navigation design the structure of the web application as well as the navigation possibilities are defined. The NAVIGATION MODEL is based on the COMCEPTUAL MODEL. Koch (2001) refers to it as a "view over the conceptual model". Designing the navigation is done in two steps. First, the navigation space model is defined. In this model the objects that should be reachable through navigation are defined. The second step is designing the navigation structure, i.e. how these objects can be reached.
Design presentation	Design presentation intends to define where and how the objects included in the navigation are presented to the user. Two models are presented to model the dynamic presentation of a web application. Object lifecycle models are used to visualize the behavior of complex presentation objects and presentation flow models show which objects are active and displayed in certain frames or windows. The PRESENTATION MODEL is visualized in UML state diagrams and UML sequence diagrams.
Design adaptation	Adaptation design consists of the definition of adaptation rules and the graphical representation of these rules in a ADAPTATION MODEL, represented by a UML collaboration model (Koch, 2001). Koch distinguishes three types of hypermedia adaptation: adaptive content, adaptive navigation support and adaptive presentation.
Design classes	The detailed design of classes is done in later iterations. Activities include defining class operations and attributes, identifying aggregation, association inheritance and dependency, describing methods, determining states and establishing requirements relevant to the implementation. The artifact is a set of descriptions of DESIGN CLASSES.
Design subsystems and interfaces	Design SUBSYSTEMS and INTERFACES. The reason behind the division in subsystems is that these subsystems can be specified and implemented by different developers. The activity results in the identification of a set of subsystems and a set of interfaces and a description of the relations to each other.

11.5. Risk management workflow

11.5.1. Introduction

The risk management workflow has as purpose to identify risks in a software development project early in the inception phase. Activities in the workflow are carried out by the project manager. The risk management workflow results in two artifacts: the risk list and the action list for risk strategy.

Koch (2001) defines risk in a software developments process as "a variable that, within its normal distribution, can take a value that endangers or reduces the success of the project". Risk management includes carrying out activities that identify, evaluate and analyze risks, and define a risk strategy in order to handle the risks. Two important parts can be distinguished in the risk management workflow: the analyze part and the action part. Both parts are continuously updated during the project. When a new risk is identified and analyzed, automatically the action list has to be updated.

11.5.2. Meta-model

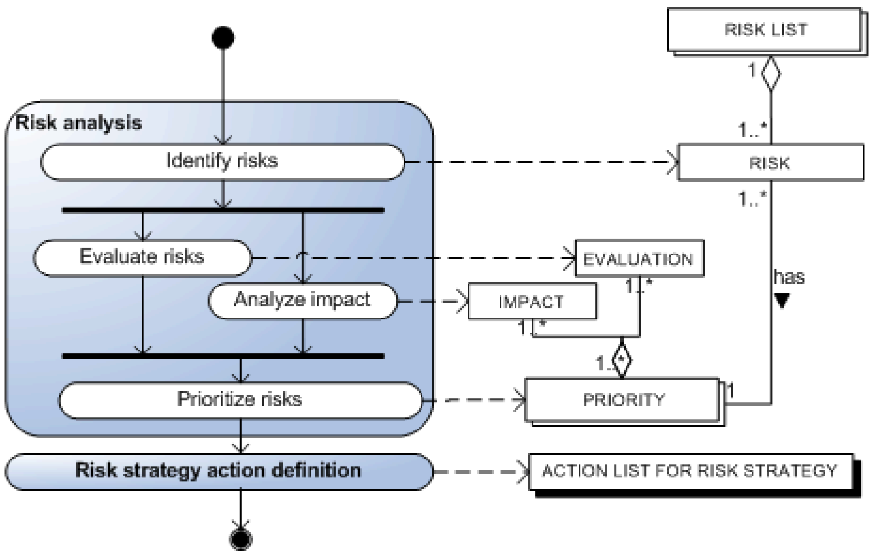


Figure 11-5: Meta-model risk management workflow

Table 11-3: Activities and sub-activities in the UWE risk management workflow

Activity	Sub-Activity	Description
Risk analysis	Identify risks	Identifying risks can be done by using standard checklists or organizing risk workshops. The RISKS are included in the RISK MANAGEMENT chapter. Koch (2001) mentions several specific risks for a web application development project, which are: ▪ use of innovative Web technologies; ▪ complexity of multimedia content, navigation structure and/or presentation; ▪ experience of the workers with the implementation of adaptive mechanisms; and ▪ difficulty of the user monitoring process.

	Evaluate risks	Every RISK is gets an EVALUATION; a description and estimation about the complexity or uncertainty of a project is given.
	Analyze impact	Analyzing the impact of a risk handles about the IMPACT a risk has on the success of a project. The evaluation and risk values can be indicated by selecting a value: low, moderate or high.
	Prioritize risks	Prioritizing the risks is done by combining IMPACT and EVALUATION in a table. High priority is then given to RISKS with the highest scores.
Define actions for risk strategy		Risk strategy actions can be obtained from experience or from relevant literature. The project manager adapts the actions to the project in the ACTION LIST FOR RISK STRATEGY. RISKS with the highest priority are on top of the list and need to be handled first.

11.6. Iteration planning workflow

11.6.1. Introduction

Iteration planning is done by the project manager. He defines initial and final states, costs, milestones and deliverables, which finally results in an iteration plan and a delivery plan.

The first iteration plan, constructed in the first iteration of the project, is a plan for the basic phases (inception, elaboration, construction, transition and maintenance). In later iterations this plan is adjusted and detailed.

11.6.2. Meta-model

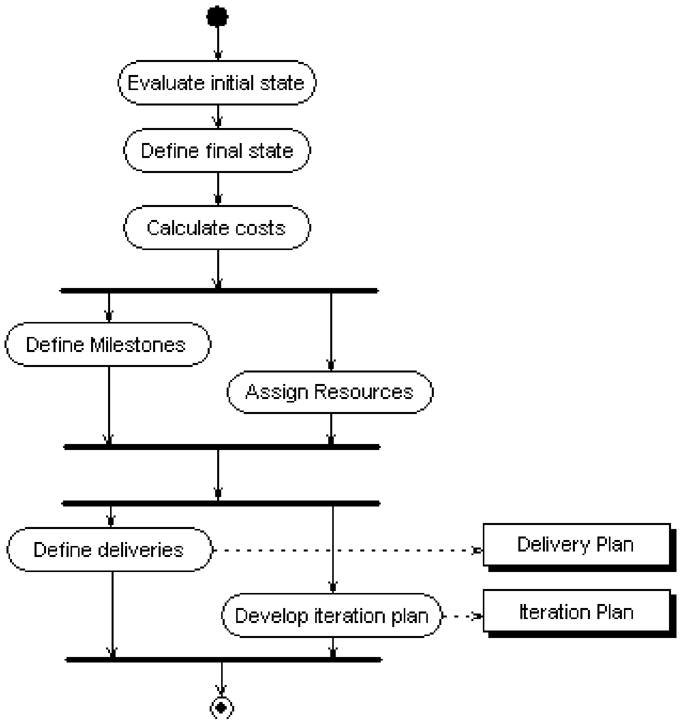


Figure 11-6: Meta-model iteration planning workflow

Table 11-4: Activities in the UWE iteration planning workflow

Activity	Description
Evaluate initial state	Describe the initial state, the state of the information system or environment at the starting point of the project. This could be, for example, a non computer-based environment.
Define final state	Describe the final state, a description of the vision of the software system to be built. This vision is also used as a foundation of the requirements capture workflow.
Calculate costs	Since calculating the costs is a difficult process, it is important to document cost estimations, in order to use this information in later projects.
Define milestones	Milestones are defined based on the activities planned and risks identified in the project.
Assign resources	Resources are assigned for each iteration.
Define deliveries	For every milestone the status of the development is described. Deliveries consist of a report with the development status and documents (e.g. the requirements analysis), models (e.g. the use case model), descriptions, packages or releases. In the DELIVERY PLAN is described what needs to be delivered at every milestone.
Develop iteration plan	In the ITERATION PLAN every phase of the development process is described. Contents are status of the project, a list of deliverables, list of risks, a list of changes, a list of validation activities, dates for reviewing the deliveries, etc.

11.7. Validation workflow

11.7.1. Introduction

The validation workflow has as purpose to check whether the result complies with what the customer wants. The architect writes an architecture review report and the use case reviewer writes a requirements review report.

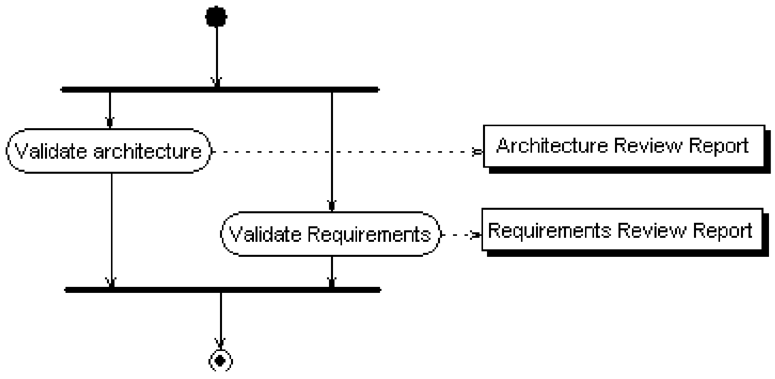


Figure 11-7: Meta-model validation workflow

Table 11-5: Activities in the UWE validation workflow

Activity	Description
Validate requirements	For validating requirements checklists described in information systems development literature can be used. A walkthrough through a prototype can be used to check the satisfaction of the requirements. The activity results in a REQUIREMENTS REVIEW REPORT.
Validate architecture	Validating the architecture focuses on error detection in the model, finding requirements that are missing in the architecture model, assessing the observation of user behavior, assessing the adaptive functionality, and avoiding architecture over-design. The activity results in a ARCHITECTURE REVIEW REPORT.

Part V: GX WebEngineering Method

In part V, the last step of the assembly-based situational method engineering process is described. This step resulted in a method, called GX WebEngineering (WEM).

In chapter 14, the method for standard implementation situations is described. Then, in chapter 15, the migration situation implementation method is outlined. In chapter 16, the method for migration situation implementation is described. Subsequently, in chapter 17, the integrated methods are shown, which form together the GX WebEngineering Method.

Chapter 18, the methods are validated in a case study. In chapter 19 conclusions are drawn, which are discussed in chapter 20. Also, recommendations on future research are given here.

12 Standard projects in WEM

12.1. Introduction

In the following sections the method for standard projects is described. Firstly, the entire process is described. Then, every project phase is described by its method rationale and meta-model. The method fragments in the acquisition phase originate from the old GX method (blue) and the Unified Process (yellow).

12.2. Process

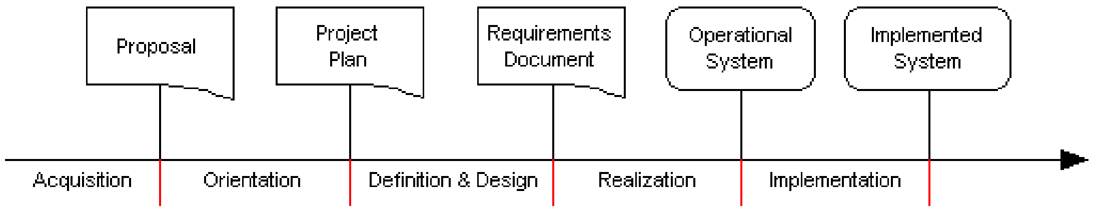


Figure 12-1: Standard project timeline

A standard project consists of five phases. The definition and design phase are integrated and result in a requirements document. The architecture document has been omitted from the process, since a standard process needs none or little software engineering to develop the website. Thus, the existing WebManager architecture is sufficient. If, nevertheless, small architecture adjustments are necessary, they are described in the requirements document. In the following the acquisition, orientation and definition & design phases are described.

12.3. Acquisition phase

12.3.1. Method rationale

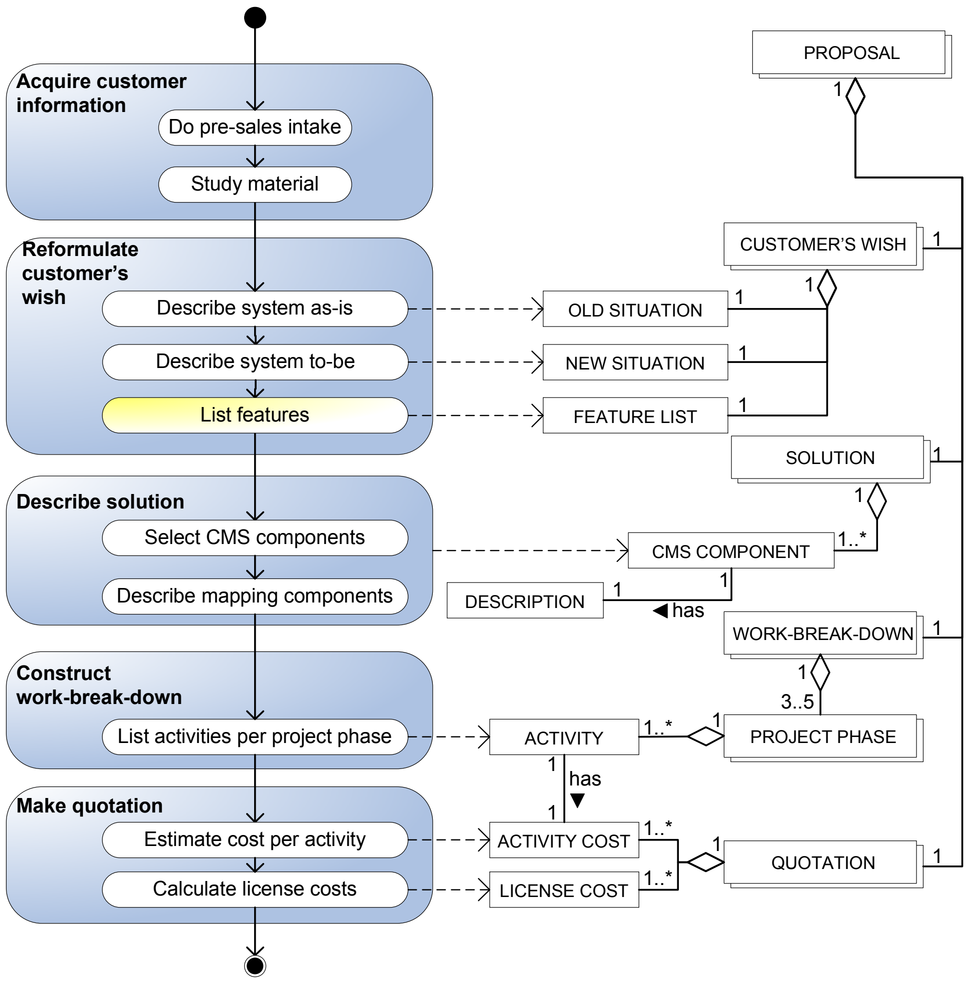
The acquisition phase is carried out by the pre-sales consultant and results in a proposal. Five main activities are recognized: *acquire customer information*, *reformulate customer's wish*, *describe solution*, *construct work-break-down* and *make quotation*.

The first problem that is tackled in this phase is the information discrepancy between pre-sales consultant and account manager. The process should start with a standard pre-sales intake of the account manager and pre-sales consultant.

The second issue is the small amount of time that is available for the acquisition phase of a standard project. The answer is describing the solution from the viewpoint of standard WebManager components, instead of the viewpoint of functionalities. These WebManager components are standard descriptions that should be reusable and centrally available. Describing the solution and constructing the work-break-down should be merely an issue of selecting the right texts, instead of describing solutions and activities. The purpose is to prevent consultant from inventing the wheel over and over again.

The method has reused most existing GX method fragments. One activity has been used from the Unified process, namely "list features". The activity is used as input for the choice which WebManager components to use. The customer benefits from it in the sense that he can see how his candidate requirements (the features) are translated to the application.

12.3.2. Meta-model



12-1: Meta-model of the acquisition phase in a standard project in WEM

Table 12-1: Activities and sub-activities in a standard acquisition phase in WEM

Activity	Sub-Activity	Description
Acquire customer information	Do pre-sales intake	Pre-sales intake is performed by account manager, consultant(s), project manager, and (possibly) software engineer. The budget is determined and customer information is exchanged.
	Study material	Material (e.g. existing website, request for proposal) is studied.
Reformulate customer's wish	Describe system as-is	Describe the OLD SITUATION to establish a basis. This basis also provides certain assumptions on, for example, the legacy systems.
	Describe system to-be	Describe the NEW SITUATION in terms of some global functionality's. It is important to be not too specific; details are described in the solution.
	List features	Ideas collected during client meetings, requirements elicitation, material study etc. are added to the FEATURE LIST. This list is used again in the definition phase.

Describe solution	Select CMS components	Select the CMS COMPONENTS that are used in the SOLUTION.
	Describe mapping components	Every component has a DESCRIPTION, which is inserted in the proposal and possibly adjusted to the particular situation.
Construct WBD	List activities per project phase	SOLUTIONS are translated into ACTIVITIES and scheduled in de WORK-BREAK-DOWN per PROJECT PHASE.
Make quotation	Estimate cost per activity	Per ACTIVITY the ACTIVITY COST is estimated. This is part of the QUOTATION.
	Calculate license costs	The LICENSE COSTS are estimated. This is part of the QUOTATION.

12.4. Orientation phase

12.4.1. Method rationale

The orientation phase of a standard project is quite similar to the existing process of the orientation phase at GX. The only change is that during a standard project no need exists to keep a detailed risk log. Three main activities can be identified: *describing the project*, *constructing the planning* and *controlling the project*. No method fragments from the Unified Process or UWE are used here.

12.4.2. Meta-model

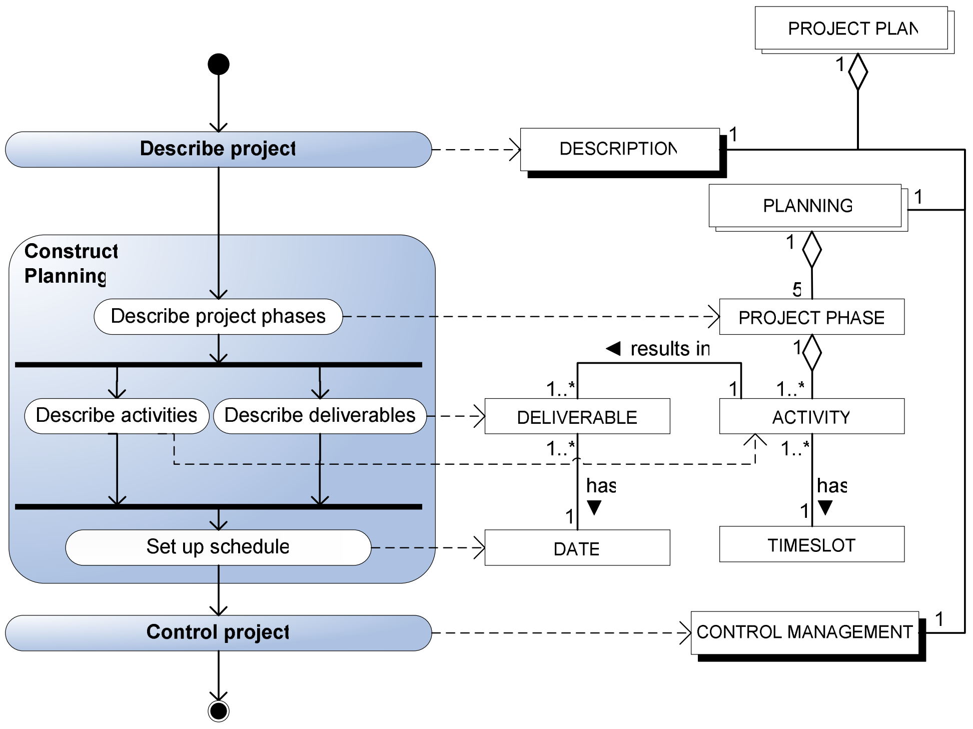


Figure 12-2: Meta-model of the orientation phase in a standard project in WEM

Table 12-2: Activities and sub-activities in a standard orientation phase in WEM

Activity	Sub-Activity	Description
Describe project		Describing the project is done in terms of participants, targets, products, scope and assumptions. This information is derived from the proposal, but with more emphasis on the project management issues. The activity end in a project DESCRIPTION.
Construct planning	Describe project phases	The PLANNING is divided into five PROJECT PHASES, which should be shortly described.
	Describe activities	De ACTIVITIES are described and grouped into PROJECT PHASES
	Describe deliverables	The DELIVERABLES that result from the ACTIVITIES are described.
	Set up schedule	For every DELIVERABLE a DATE is set and for each ACTIVITY a TIME SLOT is estimated.
Control project		Controlling the project results in a CONTROL MANAGEMENT artifact. This artifact is not further explained here, since it concerns regular project management issues, like communication management, progress management, change management and problem management that lie outside the scope of this research.

12.5. Definition

12.5.1. Method Rationale

The method fragments in the definition phase originate from the old GX method (blue) and UWE (red). The first activity, *goalsetting,* is intended to inform the reader of the background, scope, assumptions etc. of the project.

During the definition phase the requirements are not described in terms of use cases. Use case modeling appeared to take to much time for the relatively small budget standard projects have. Therefore the requirements are described per category. In *user and domain modeling* the visitor and WebManager user needs are described, as well as the information need of the visitor. This activity originates from UWE. During the *application modeling* functional and non-functional requirements are described, resulting in a website model and application model. This activity also originates from UWE.

Finally, the *additional requirements are described.* GX uses an adapted categorization of IEEE (1998), which is also used here. It consists of: support, security, system performance, scalability, interface and design conditions.

12.5.2. Meta-model

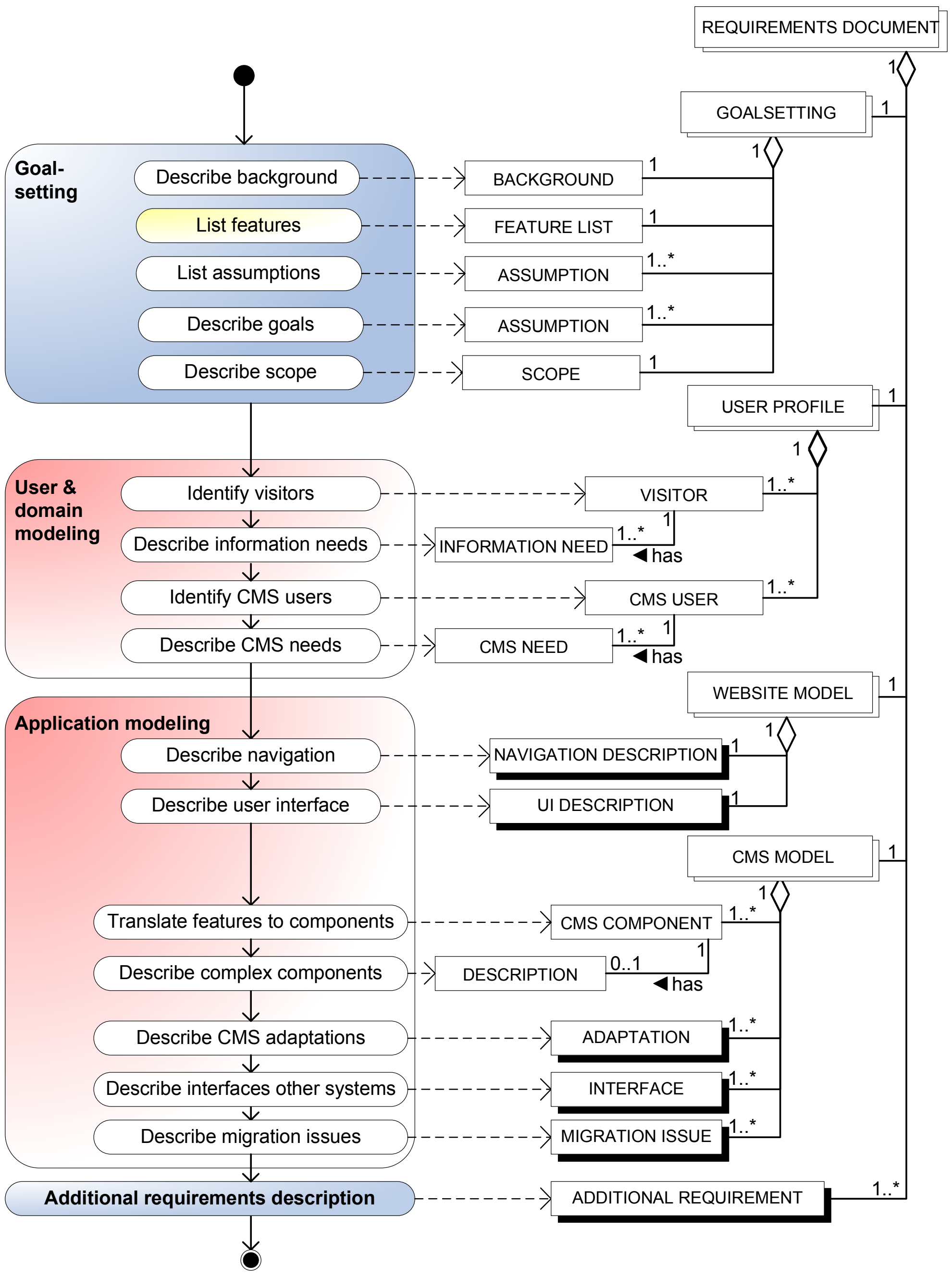


Figure 12-3: Meta-model of the definition phase in a standard project in WEM

Table 12-3: Activities and sub-activities in a standard definition phase in WEM

Activity	Sub-Activity	Description
Formulate product vision	Describe background	Describe the context of the project, its business drivers etc. in BACKGROUND.
	List features	Extend the FEATURE LIST described in the acquisition phase with features that come up during the definition phase.
	List assumptions	List the ASSUMPTIONS you use in the rest of the document.
	Describe goals	Describe the business GOALS the customer want to achieve with this project.
	Describe scope	Describe the SCOPE of the project.
User and domain modeling	Identify visitors	Identify the VISITORS of the front-end of the web application, for example member, customer, etc.
	Describe information needs	Describe the INFORMATION NEED per VISITOR.
	Identify CMS users	Identify CMS USERS, for example editor, publisher, etc.
	Describe CMS needs	Describe the NEEDS per CMS USER.
Application modeling	Describe navigation	Describe the NAVIGATION of the front-end of the web application. This can be done by, for example, scenarios, schemas or screenshots.
	Describe user interface	Describe the USER INTERFACE. This can be done by showing prototype screens (if available), or by describing user interface guidelines.
	Translate features to components	List the features describes in the FEATURE LISTS in a table, together with the COMPONENTS they are translated to.
	Describe complex components	If necessary, give a DESCRIPTION of the complex components listed in the COMPONENT table.
	Describe CMS adaptations	Describe ADAPTATIONS that have been made to the CMS.
	Describe interfaces with other systems	When the web application interacts with other systems, the INTERFACES between these systems should be described here.
	Describe migration issues	In case of a MIGRATON of existing content to the new web application, migration issues like format, quantity etc. should be described here.
Additional requirements description		Describe ADDITIONAL REQUIREMENTS concerning aspects like performance, security, and maintainability of the system.

13 Complex projects in WEM

13.1. Introduction

In the following the method for complex projects is described. Firstly, the entire process is described. Then, every project phase is described by its method rationale and meta-model.

13.2. Process

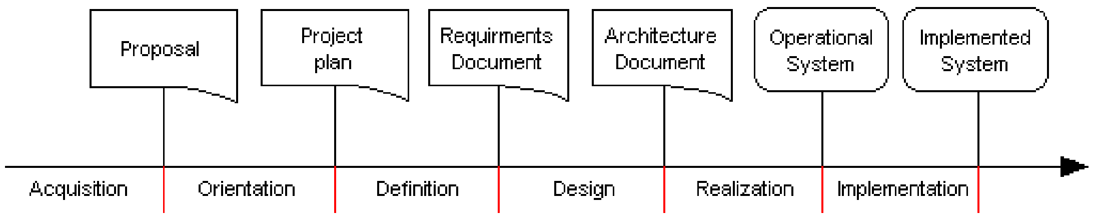


Figure 13-1: Complex project timeline

A complex project consists of six phases, each ending in a deliverable. In the next sections the first three phases are described, which are the acquisition, orientation and definition phases. The other phases are outside the scope.

In a complex project several artifacts are added compared to a standard project. A complex project always delivers a requirements document in the definition phase. A requirements document delivers the wishes of the customer. Typical things that belong in a requirements document are domain models, use cases and workflow descriptions. The architecture document has the purpose to define the software architecture of the web application.

13.3. Acquisition phase

13.3.1. Introduction

The acquisition phase is carried out by the account manager, pre-sales consultant and project manager. The process modeling starts at the pre-sales intake and results in a proposal, written by the pre-sales consultant. The developed method is assembled from method fragments of GX and the Unified Process.

13.3.2. Method rationale

The method fragments in the acquisition phase originate from the old GX method (blue), UWE (red), and the Unified Process (yellow).

Five main activities can be recognized in the acquisition phase. The acquisition phase begins with *acquiring customer information*, which starts with a pre-sales intake with the account manager, pre-sales consultant and project manager. In the pre-sales intake is decided whether the project can start by simply studying the available material, or that an extensive requirements elicitation process is needed in order to handle the complexity of the project. This method fragment is an adjusted version of an existing GX method fragment and part of an UWE fragment.

Secondly, in *reformulating the customer's wish* the customer's wish (as explained by the customer), assumptions and scope are described. Also, based on the studied material and possibly the extensive requirements elicitation, the main functionalities are described. This method fragment is an adjusted version of an existing GX method fragment.

During the third main activity, *listing the candidate requirements*, ideas of the customer and consultant are arranged in a feature list, which was already created in the acquisition phase. Input for this feature list is collected during client meetings, requirements elicitation, material study and other contacts with the customer. During the whole development process features can be added and evaluated. This method fragment is adopted from the Unified Process.

Then, in *describing the solution* standard solutions and custom solutions are described. This division is made to make clear to the customer and to GX which solutions can be provided by standard WebManager components and which solutions need software engineering. This method fragment is an adjusted version of an existing GX method fragment.

Finally, *constructing a work-break-down* and *making a quotation* are done in the same way as in a standard project. The only difference is that the consultant informs the software engineer more extensively, especially about the custom solutions. Both method fragments are existing GX method fragments.

13.3.3. Meta-model

In Figure 13-2 the meta-model of the acquisition phase is illustrated. The meta-model is explained in Table 13-1. Sub-activities are grouped into activities and each activity is provided with a description. In each description is indicated which artifact results from that activity.

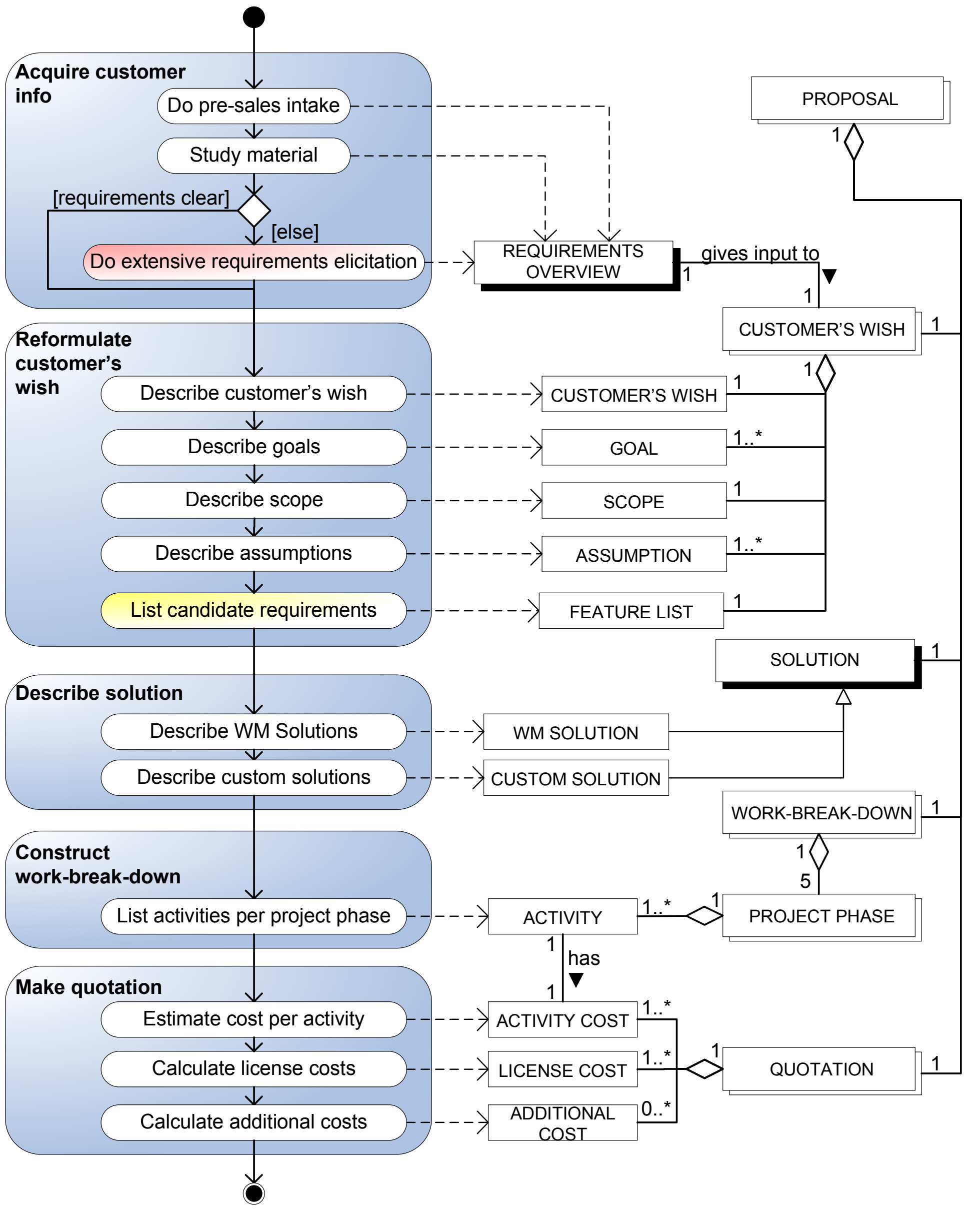


Figure 13-2: Meta-model of the acquisition phase in a complex project in WEM

Table 13-1: Activities and sub-activities in a complex acquisition phase in WEM

Activity	Sub-Activity	Description
Acquire customer information	Do pre-sales intake	Pre-sales intake is performed by account manager, consultant(s), project manager, and (possibly) software engineer. The budget is determined and customer information is exchanged.
	Study material	Material (existing website, RFP etc.) is studied.
	Requirements branch	*Based on the intake and the studied material, the decision is taken whether to do an extensive requirements elicitation or not.*
	Do extensive requirements elicitation	Extensive requirements elicitation can be done by different techniques like interviewing, workshops, brainstorming etc.

Reformulate customer's wish	Describe customer's wish	The customer's WISH is described (shortly).
	Describe assumption	ASSUMPTIONS, for example about legacy systems, are described.
	Describe scope	SCOPE is described to make clear to the customer and to GX what is done by GX and what is not.
	Describe main functionalities	Based on preceding activities the MAIN FUNCTIONALITIES are described.
	List candidate requirements	Ideas collected during client meetings, requirements elicitation, material study etc. are added to the FEATURE LIST.
Describe solution	Describe WM solution	WM SOLUTIONS are described, based on the FEATURE LIST .
	Describe custom solutions	CUSTOM SOLUTIONS are described, based on the FEATURE LIST.
Construct WBD	List activities per project phase	SOLUTIONS are translated into ACTIVITIES and scheduled in de WORK-BREAK-DOWN per PROJECT PHASE.
Make quotation	Estimate cost per activity	Per ACTIVITY the ACTIVITY COST is estimated. This is part of the QUOTATION.
	Calculate license costs	The LICENSE COSTS are estimated. This is part of the QUOTATION.
	Calculate additional costs	ADDITIONAL COSTS, if applicable, are calculated and added to the QUOTATION.

13.4. Orientation phase

13.4.1. Introduction

The project manager carries out the orientation phase, which results in a project plan. The developed method is assembled from method fragments of GX and UML-based Web Engineering.

13.4.2. Method rationale

Four main activities can be recognized in the acquisition phase. The method fragments in the acquisition phase originate from the old GX method (blue) and UWE (red).

The acquisition phase begins with *describing the project*, in order to inform the reader. This method fragment is adopted form the GX method.

Secondly, in *constructing the planning* a work-break-down in terms of phases, activities, deliverables and dates is provided. This method fragment is adopted form the GX method.

During the third main activity, *controlling the project*, issues like organization, communication management, progress management, change management and problem management are described. This method fragment is adopted form the GX method.

Finally, *control risk* has as purpose to identify risks in a software development project early in the inception phase. This method fragment, as well as the action strategy that is defined based on this risk control, is adopted from UWE.

13.4.3. Meta-model

In Figure 13-3 the meta-model of the orientation phase is illustrated. The meta-model is explained in Table 13-2. Sub-activities are grouped into activities and each activity is provided with a description. In each description is indicated which artifact results from that activity.

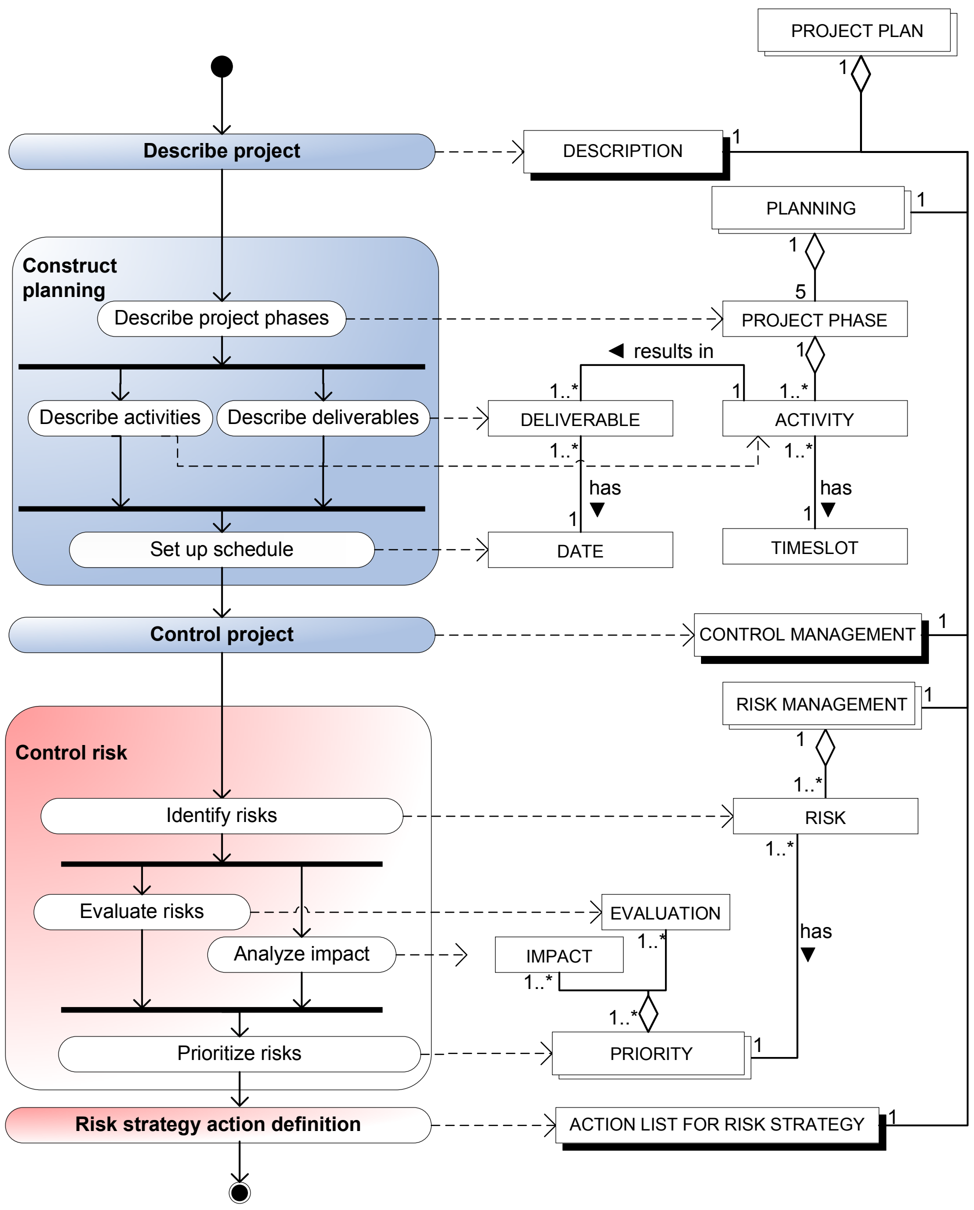


Figure 13-3: Meta-model of the orientation phase in a complex project in WEM

Table 13-2: Activities and sub-activities in a complex orientation phase in WEM

Activity	Sub-Activity	Description
Describe project		Describing the project is done in terms of participants, targets, products, scope and assumptions. This information is derived from the proposal, but with more emphasis on the project management issues. The activity end in a project DESCRIPTION.
Construct planning	Describe project phases	The PLANNING is divided into five PROJECT PHASES, which should be shortly described.
	Describe activities	The project ACTIVITIES are described and grouped into PROJECT PHASES
	Describe deliverables	The DELIVERABLES that result from the project ACTIVITIES are described.
	Set up schedule	For every DELIVERABLE a DATE is set and for each ACTIVITY a TIME SLOT is estimated.
Control project		Controlling the project results in a CONTROL MANAGEMENT artifact. This artifact is not further explained here, since it concerns regular project management issues, like communication management, progress management, change management and problem management that lie outside the scope of this research.
Control risk	Identify risks	Identifying risks can be done by using standard checklists or organizing risk workshops. The RISKS are included in the PROJECT PLAN.
	Evaluate risks	Every RISK is provided with an EVALUATION; a description and estimation about the complexity or uncertainty of a project is given.
	Analyze impact	Analyzing the impact of a risk handles about the IMPACT, a risk has on the success of a project. The evaluation and risk values are indicated by selecting a value: low, moderate or high.
	Prioritize risks	Prioritizing the risks is done by combining IMPACT and EVALUATION in a table. High priority is then given to RISKS with the highest scores.
	Define actions for risk strategy	Risk strategy actions can be obtained from experience or from relevant literature. The project manager adapts the actions to the project in the ACTION LIST FOR RISK STRATEGY. RISKS with the highest priority are on top of the list and need to be handled first.

13.5. Definition phase

13.5.1. Introduction

The definition phase is mainly carried out by the consultant(s). The modeled process starts with requirements elicitation and result in a requirements analysis document and a requirements review report. The developed method is assembled from method fragments from GX, the Unified Process and UML-base Web Engineering.

13.5.2. Method rationale

The method fragments in the acquisition phase originate from the old GX method (blue), UWE (red), and the Unified Process (yellow). Seven main activities can be recognized in the definition phase. First of all, an *extensive requirements elicitation* is performed. Based on the findings of this process, the requirements are described. This activity originates from UWE.

Then, the requirements analysis is based on *the product vision* of the web application to be built. In the perfect situation, the client writes this product vision himself, identifying targets, assumptions and scope. However, in practice the consultant writes this document, after acquiring information in cooperation with the customer. In line with the acquisition phase, the features are updated here. This sub-activity originates from the Unified Process.

Thirdly, *domain modeling* is used to create a common ground; that is, the terms with its definitions and relations need to be defined, in order to come to a conceptual object model. This domain model is the base of all other activities.

Fourthly, the *use case modeling* starts. In a use case diagram all actors and their functions (the use cases) are described. This fragment originates from the Unified Process. A distinction is made between use cases that can be captured in standard WebManager components and custom use cases.

The fifth activity, *application modeling*, is described what kind of web application needs to be developed, in terms of user interface, navigation etc. This method fragment originates from UWE.

Finally, the requirements need to be validated. This can be done by using existing checklists, or by performing a walkthrough. This method fragment also originates from UWE.

13.5.3. Meta-model

In Figure 13-3 the meta-model of the definition phase is presented. The meta-model is explained in Table 13-2. Sub-activities are grouped into activities and each activity is provided with a description. In each description is indicated which artifact results from that activity.

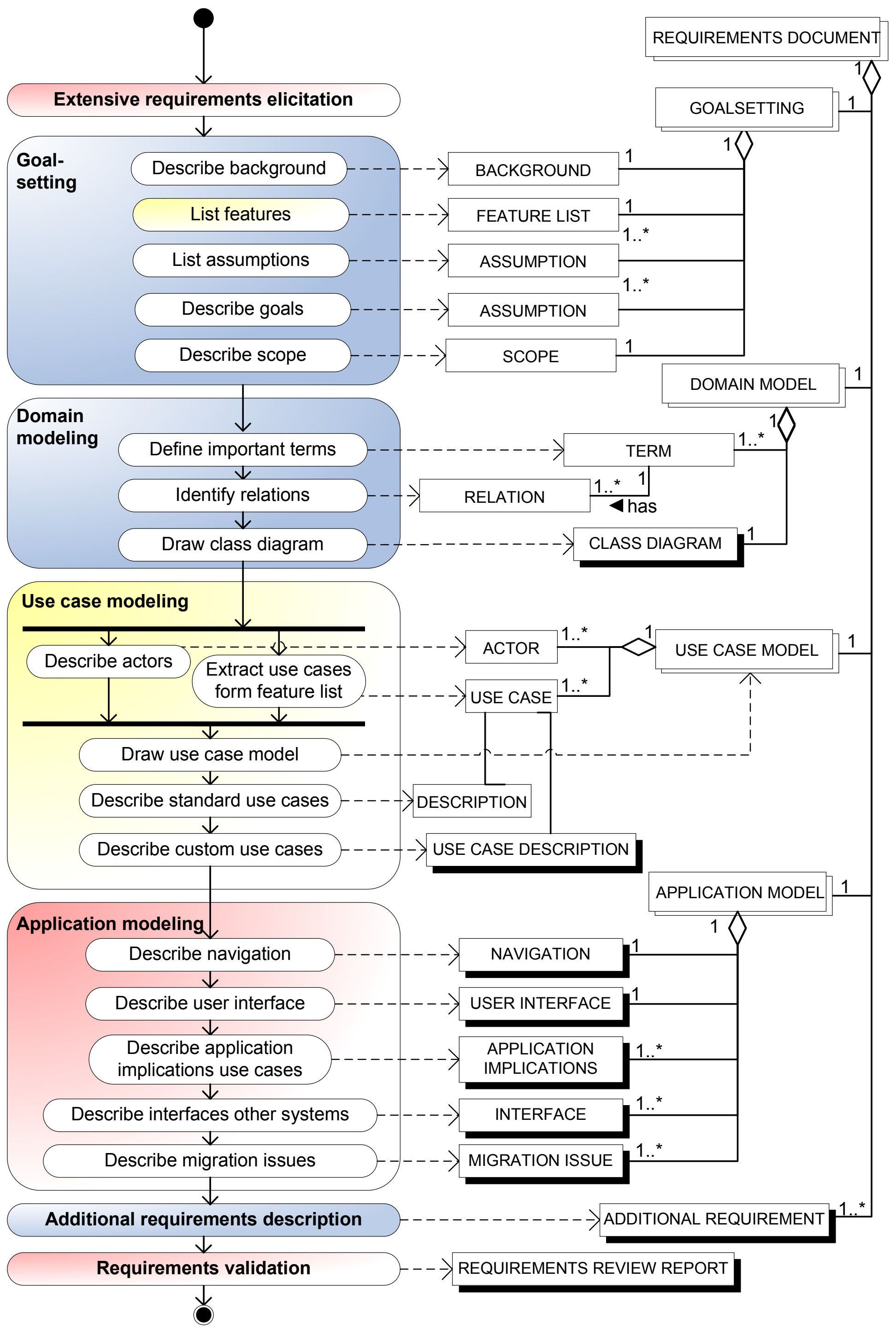


Figure 13-4: Meta-model of the definition phase in a complex project in WEM

Table 13-3: Activities and sub-activities in a complex definition phase in WEM

Activity	Sub-Activity	Description
Do extensive requirements elicitation		Extensive requirements elicitation can be done by different techniques like interviewing, workshops, brainstorming etc.
Formulate product vision	Describe background	Describe the context of the project, its business drivers etc. in BACKGROUND.
	List features	Extend the FEATURE LIST described in the acquisition phase with features that come up during the definition phase.
	List assumptions	List the ASSUMPTIONS you use in the rest of the document.
	Describe goals	Describe the business GOALS the customer wants to achieve with this project.
	Describe scope	Describe the SCOPE of the project.
Domain modeling	Define important terms	Define all important TERMS that are used in the requirements document.
	Identify relations	Identify RELATIONS between the defined TERMS.
	Draw conceptual object model	Make an overview TERMS and RELATIONS in the CLASS DIAGRAM.
Use-case modeling	Describe actors	Describe the involved ACTORS
	Extract use cases from feature list	Translate features from the FEATURE LIST into USE CASES.
	Develop use case model	Structure ACTORS and USE CASES in a USE CASE MODEL.
	Describe us -case model	Describe the USE CASE MODEL.
	Describe standard use cases	Provide the USE CASES marked as standard CMS type with a standard DESCRIPTION, possibly adapted to the specific situation.
	Detail custom use cases	Provide the USE CASES marked as custom type with a USE CASE DESCRIPTION.
Application modeling	Describe navigation	Describe the NAVIGATION of the front-end of the web application. This can be done by, for example, scenarios, schemas or screenshots.
	Describe user interface	Describe the USER INTERFACE. This can be done by showing prototype screens (if available), or by describing user interface guidelines.
	Describe application implications use cases	The USE CASES that have implications on the web application that cannot be described in the USE CASE DESCRIPTION, but are important, are described here.
	Describe interfaces other systems	When the web application interacts with other systems, the INTERFACES between these systems should be described here.
	Describe migration issues	In case of a MIGRATON of existing content to the new web application, migration issues should be described here.
Additional requirements description		Describe ADDITIONAL REQUIREMENTS concerning aspects like performance and maintainability of the system.
Requirements validation		Validate the requirements by using existing checklists, or by performing a walkthrough. This validation results in a REQUIREMENTS REVIEW REPORT.

14 Migration Projects in WEM

14.1. Introduction

In the following, the method for migration projects is described. UWE and the Unified Process give little attention to migration projects. No specific method for the content of migrating web applications exists. However, relevant research has been done in the field of information system migrations. This chapter starts with an overview of two articles, which can give important input on how to do a migration project. After the literature overview, the entire migration process is described. Finally, every project phase is described by its method rationale meta-model.

14.2. Literature

In scientific literature the problems that occur with migrations of information systems are acknowledged. Several academics and professionals try to find a solution on how to keep a migration project within time and budget. In the following subsections, two articles from different points of view are described. The first one is a scientific article about migrating legacy information systems and the second one is an article about migrations from CMSwatch.com.

14.2.1. Legacy Information System Migration: A Brief Review of Problems, Solutions and Research Issues

Bisbal, Lawless, Wu and Grimson (1999) identify five phases in a migration process. The first phase is *justification*. This is important, since migrations often are expensive projects. The costs and benefits should be weighed against each other to justify the acquisition of a new WebManager version. The second phase is *legacy system understanding*. Bisbal et al. (1999) stress the importance of understanding the functionality and domain interaction of the legacy system. It is important to ascertain which functions already meet the requirements of the user. A poor understanding of the existing system leads to an incorrect requirements specification. Besides understanding the legacy applications, the structure of the legacy data should be understood as well. *Target system development* is the third phase in the migration project. In this phase the requirements specification is prepared. This can only be done after the legacy system is understood. The architecture should facilitate maintenance and extension of the system in the future. The fourth phase is *testing*. Testing should be done without the new functionalities incorporated in the new system. After the migration and testing of the web application, new functionalities can be implemented. The last phase is the actual *migration*. Three transition strategies are described:

- Cut-and-run strategy, which consists of a transition from the old to the new system in one single step.
- Phased interoperability, which describes a cut-over of small incremental steps and every step replaces a few legacy components.
- Parallel operations, where the legacy and new system are both kept online until the new system is finished and the legacy system can be shut down.

14.2.2. CMSWatch.com: Web Content Migration Project Design

CMSWatch.com provides an independent source of information, trends, opinion, and analysis about Web Content Management and Enterprise Content Management solutions (Haniph, 2004). They published an article about the process to effectively and efficiently migrate web content. The process is divided into three parts, each handling several issues. In the following these issues are described.

Assembling the team
Before the actual start of a migration project, a team has to be assembled. It is important to *assemble a cross-functional team*. This means that the team should at least contain an "experienced project manager with executive support that can negotiate across business units". Two technical persons are needed, one with knowledge of the old web application and one with knowledge of the new web application. Early in the process is it necessary to *reframe perceptions* of the project members. It appears that these perceptions are in general too optimistic about time, resources and finances. The migration activities should be kept centralized. This is the best way to reuse and capture migration knowledge, for example knowledge from earlier migrations, automation tools etc. The last issue to keep in mind when assembling a team, is to *use the content owners wisely*. Often in migration projects, it is assumed that the content owners can help with migrating, cleaning up or changing the content. However, content owners usually do not have the time for this since they are occupied with their everyday job. What they can do is reviewing the migrated content and assuring this is accurate.

Understanding Your Content
The first step in understanding the content is *auditing your content*. According to Haniph (2004), companies only need to spend two weeks on the content audit for a site with under 4,000 pages. The second step is to *establish a content convention*. The reason for this is to make the communication more transparent. Problems have been reported on, for example, counting the content. Because, what has to be counted? Pages can be counted, or words, or paragraphs etc. To *keep track of the content*, audit trails need to be established. This allows you to be pro-active when problems occur in the migration process. The last step is to *decide what you are going to migrate*. Several content types need a special treatment, for example, highly secure content or contracts.

Delivering Results
It is important to first *run a pilot*. The pilot enables you to make estimations about the actual migration. Statistics that can be used for this estimation are:

- Speed: time to migrate a page, site freeze duration, customization time (site navigation, templates, workflows)
- Quality: pages removed, internal and external link integrity, metadata coverage
- Cost: cost per migrated page (automated), cost per migrated page (manual)
- Volume: number of pages, pages changed since latest migration (Haniph, 2004)

The second issue is the importance of *leveraging technology*. Technological solutions are faster and better than manual solutions. Only in extremely simplified cases one should consider manual migration. Thirdly, when doing the actual migration, you should *focus on mini projects*. "A "mini-site" is a self-contained grouping of content, frequently represented by a business unit, government agency or corporate function" (Haniph, 2004). A mini-site should be 2000-4000 pages in case of an automated solution and under 500 pages in case of a manual solution. The fourth issue concerns the *careful use of site freezes*. Site freezes should be kept as short as possible. Good preparations like pre-migration of content can reduce site freezes. Finally, the migration team should *develop a menu of options*. This menu of service options and prices should be provided to the customer, so they understand what they are paying for.

14.3. Process

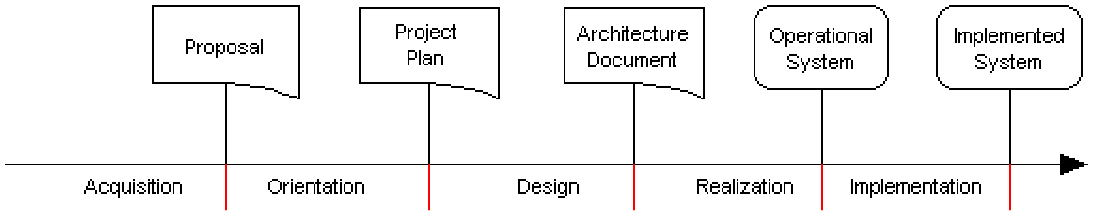


Figure 14-1: Migration Project Timeline

The process of a migration project consists of five phases. In the following the acquisition and orientation phases are described.

14.4. Acquisition Phase

14.4.1. Method Rationale

The acquisition phase is carried out by the pre-sales consultant and results in a proposal. The method fragments in the acquisition phase originate from the old GX method (blue) and fragments that originate from the described migration literature (green), namely Haniph (2004). The blue fragments are also influenced by the migration guidelines that is used at GX.

Five main activities can be recognized: *acquire customer information*, *reformulate customer's wish*, *describe solution*, *construct work-break-down* and *make quotation*.

The first activity differs slightly from other proposals. Migrations have specific characteristics that have to be addressed. Three important problems can be identified in the acquisition phase, which are:

1. The decision whether to use a migration script or migrate the content manually often turned out wrong.
2. Old WebManager installations are sometimes badly documented.
3. The client does not always know what to expect from a migration.

The first two problems are related to each other. After all, the documentation of the existing WebManager version is important by making the decision whether to use a migration script, or to manually migrate the website. To overcome these problems it is important to perform a legacy system scan by a software engineer. According to Bisbal et al. (1999) and Haniph (2004) it is important to have a thorough knowledge of the legacy systems. Legacy systems in this case include (a) the existing WebManager version, (b) the existing systems that have to be connected to the new WebManager version, (c) the structure of the existing content, and (d) the content type. The existing WebManager can be studied by reading the documentation of the project. However, it seems that this documentation often does not have the required quality. Therefore, a thorough overview of the application by a software engineer is necessary.

The second activity, *reformulating the customer's wish*, differs from a standard project in the sense that it is necessary to do a migration justification. The customer has to be explained why a newer version of WebManager is necessary and which benefits it brings.

In *describing the solution*, the problem of bad expectations from the customer is tackled. One should explain the system to-be not only in terms of new functionalities, but also in terms of changing functionalities.

14.4.2. Meta-Model Acquisition Phase

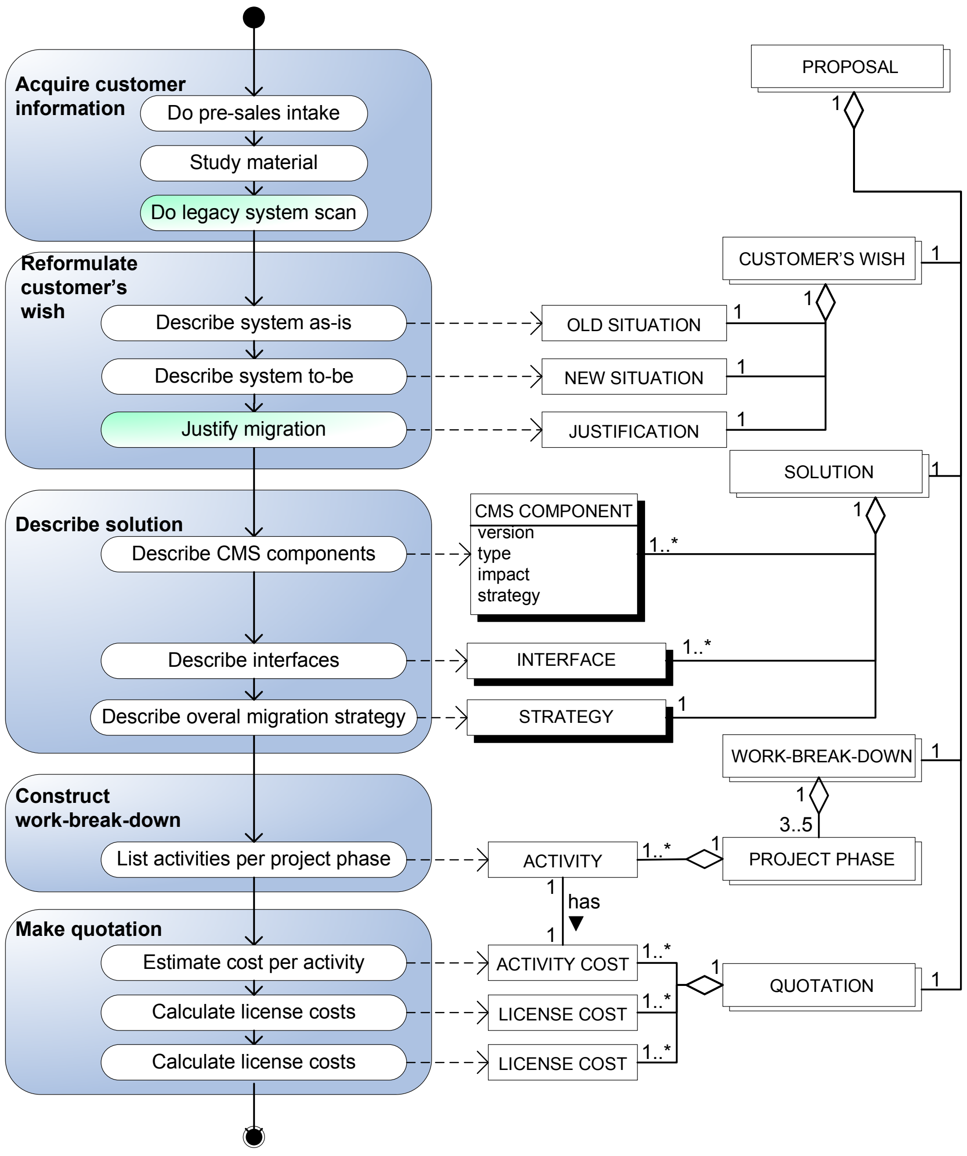


Figure 14-2: Meta-Model the acquisition phase in a migration project in WEM

Table 14-1: Activities and sub-activities in a migration acquisition phase in WEM

Activity	Sub-Activity	Description
Acquire customer information	Do pre-sales intake	Pre-sales intake is performed by account manager, consultant(s), project manager and software engineer. The budget is determined and customer information is exchanged.
	Study material	Material is studied. Important issues are: (a) How much content needs to be migrated? (b) Which version needs to be migrated? (c) How is the content structured? (d) How much customization is done in the existing WebManager version?
Reformulate customer's wish	Describe system as-is	Describe the OLD SITUATION to establish a basis. This basis also provides certain assumptions on, for example, the legacy systems.
	Describe system to-be	Describe the NEW SITUATION in terms of some global functionality's. It is important to be not too specific; details are described in the solution.
	Justify migration	In the JUSTIFICATION is explained why a new version of WebManager is needed to support the new web application.
Describe solution	Describe CMS components	Describe the CMS COMPONENTS that are used in the SOLUTION. Indicate in "version" whether it consists an old CMS component that needs to be migrated, or a new CMS component. With "type" one can indicate whether it is a standard, custom, design or content component. Finally, indicate the impact and define a strategy with the customer.
	Describe interfaces	Describe how to migrate existing INTERFACES with external systems.
	Describe overall migration strategy	Describe, together with the project manager and software architect, the overall migration strategy, based on the described strategy of the CMS COMPONENTS.
Construct WBD	List activities per project phase	SOLUTIONS are translated into ACTIVITIES and scheduled in de WORK-BREAK-DOWN per PROJECT PHASE.
Make quotation	Estimate cost per activity	Per ACTIVITY the ACTIVITY COST is estimated. This is part of the QUOTATION.
	Calculate license costs	The LICENSE COSTS are estimated. This is part of the QUOTATION.

Describing the solution consists of two activities. Firstly, the WebManager components that are new in the system to-be are described. Then, the functionalities that are changed in the new WebManager version are explained.

Constructing the work-break-down and *making the quotation* are done in the same way as a standard project. The only difference is that the pre-sales consultant closely cooperates with the software engineer and project manager.

14.5. Orientation

14.5.1. Method Rationale

Project management is an important issue in migration projects. A project management issue that is especially important in migration projects is expectation management. The customer and project team generally underestimate the project in terms of time, costs and resources. A good preparation to overcome this problem is necessary. That is the reason why the orientation phase, and thus writing the project plan, starts at the same time as the acquisition phase.

Three main activities are identified: *describing the project*, *constructing the planning* and *controlling the project*. The method is more or less the same as the method in the orientation phase of a complex project. It describes the process of developing a project plan. However, two important things, that are not modeled in the method, but should be mentioned, are the following guidelines:

1. Assemble a cross-functional team, with at least (a) an experienced project manager, (b) a software engineer with knowledge of the existing WebManager implementation, (c) a software engineer with knowledge of the new WebManager version.
2. Reframe perceptions of the project members from both GX and the customer organization.

A change in the method is setting up a content convention. This is necessary to avoid problems on conventions. An example is problems around counting the content. A convention has to be set up where is described whether the pages, words or paragraphs should be counted. This sub-activity originates from Haniph (2004). The blue fragments are influenced by the migration guidelines that is used at GX.

14.5.2. Meta-Model

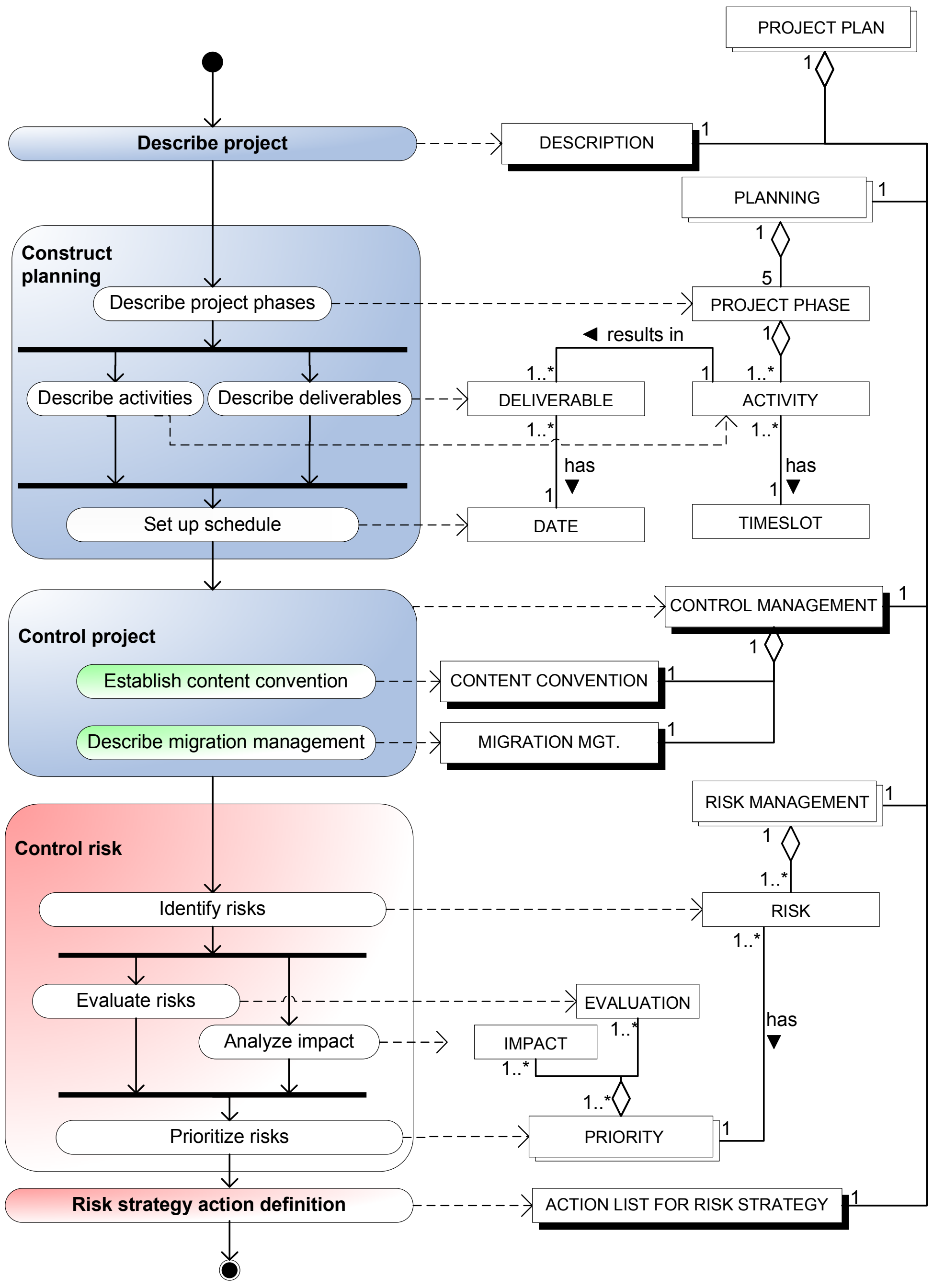


Figure 14-3: Meta-model of the orientation phase in a migration project in WEM

Table 14-2: Activities and sub-activities in a migration orientation phase in WEM

Activity	Sub-Activity	Description
Describe project		Describing the project is done in terms of participants, targets, products, scope and assumptions. This information is derived from the proposal, but with more emphasis on the project management issues. The activity end in a project DESCRIPTION.
Construct planning	Describe project phases	The PLANNING is divided into five PROJECT PHASES, which should be shortly described.
	Describe activities	De ACTIVITIES are described and grouped into PROJECT PHASES
	Describe deliverables	The DELIVERABLES that result from the ACTIVITIES are described.
	Set up schedule	For ever DELIVERABLE a DATE is set and for each ACTIVITY a TIME SLOT is estimated.
Control project		Controlling the project results in a CONTROL MANAGEMENT artifact. Only one sub-activity is described here, since the rest concerns regular project management issues, like communication management, progress management, change management and problem management that lie outside the scope of this research.
	Establish content convention	Establish the CONTENT CONVENTION to make communication more transparent.
	Describe migration management	Describe specific MIGRATION MANAGEMENT issues, like the management of two CMSs at the same time.
Control risk	Identify risks	Identifying risks can be done by using standard checklists or organizing risk workshops. The RISKS are included in the RISK MANAGEMENT chapter.
	Evaluate risks	Every RISK is provided with an EVALUATION; a description and estimation about the complexity or uncertainty of a project is given.
	Analyze impact	Analyzing the impact of a risk handles about the IMPACT a risk has on the success of a project. The evaluation and risk values can be indicated by selecting a value: low, moderate or high.
	Prioritize risks	Prioritizing the risks is done by combining IMPACT and EVALUATION in a table. High priority is then given to RISKS with the highest scores.
	Define actions for risk strategy	Risk strategy actions can be obtained from experience or from relevant literature. The project manager adapts the actions to the project in the ACTION LIST FOR RISK STRATEGY. RISKS with the highest priority are on top of the list and need to be handled first.

15 Integrated Methods

15.1. Introduction

In the preceding chapter the GX WebEngineering method was described per implementation situation. In this chapter, the methods are integrated into activity-diagrams. This means that the data-part is omitted. Route maps are provided to show the different routes that can be taken in the method, depending on the implementation situation. Furthermore, in the diagrams is indicated from which candidate method the method fragments originate.

15.2. Acquisition phase

The method fragments in the acquisition phase originate from the old GX method (blue), UWE (red), and the Unified Process (yellow). Furthermore, in the migration route fragments are included that originate of the described migration literature, namely Bisbal, Lawless, Wu and Grimson (1999) and Haniph, (2004).

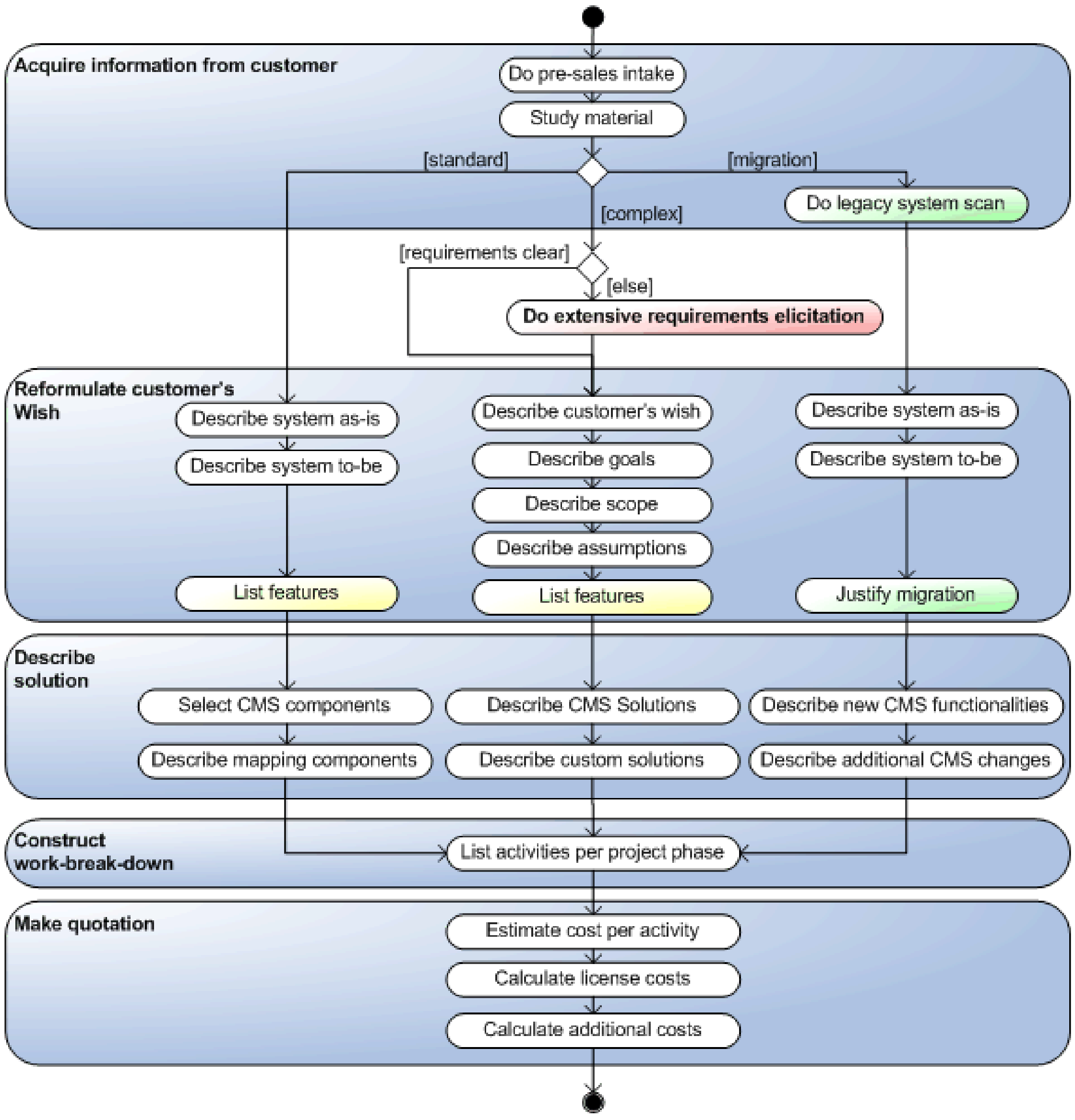


Figure 15-1: GX WebEngineering - Acquisition phase in WEM

15.3. Orientation phase

The method fragments in the acquisition phase originate from the old GX method (blue) and UWE (red). Furthermore, in the migration route a fragment is included that originates from the described migration literature, namely Haniph (2004).

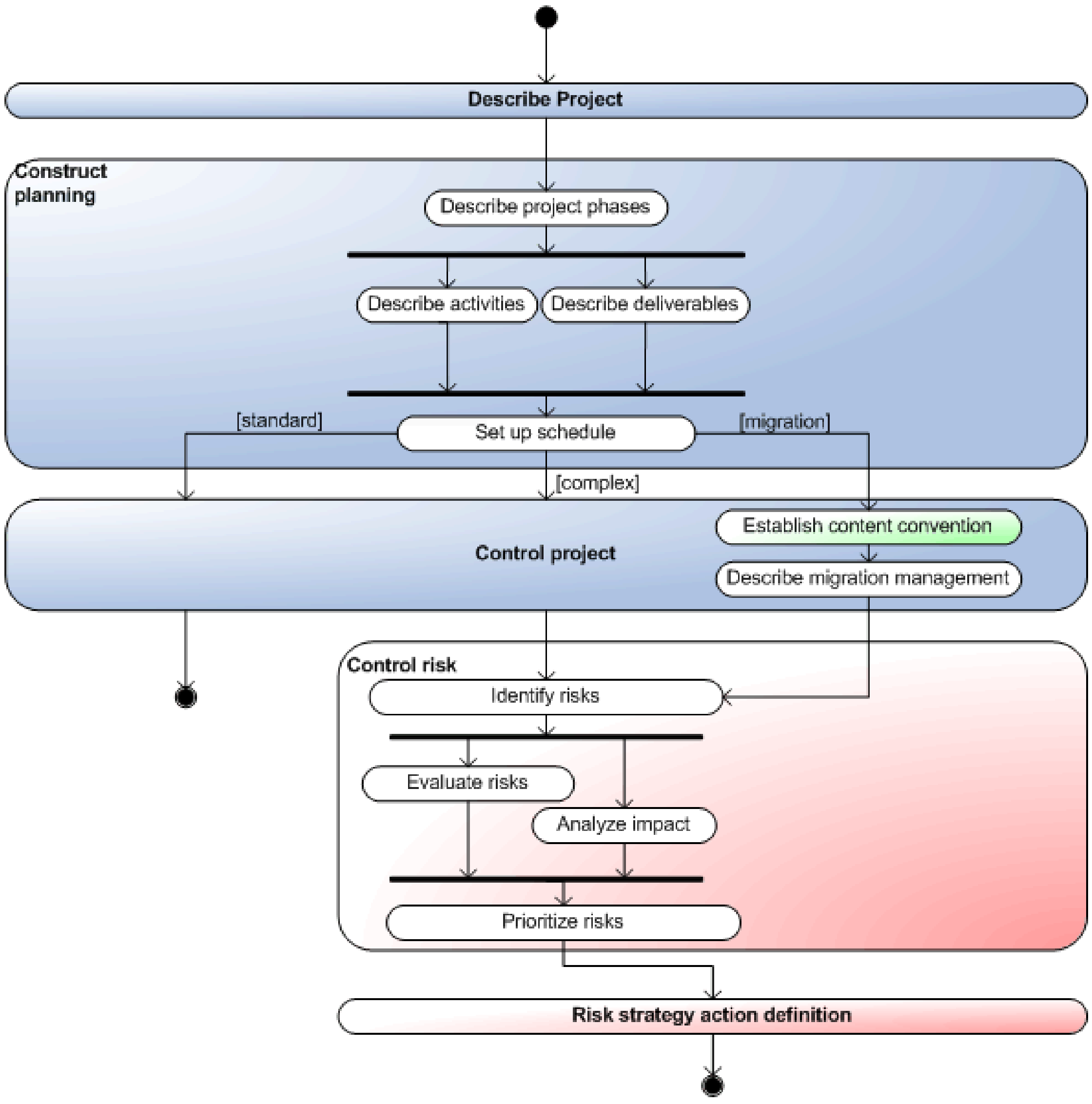


Figure 15-2: GX WebEngineering - Orientation phase in WEM

15.4. Definition phase

The definition phase is only used in standard and complex implementation situations. In migration implementation situations this phase is omitted. The method fragments in the acquisition phase originate from the old GX method (blue), UWE (red), and the Unified Process (yellow).

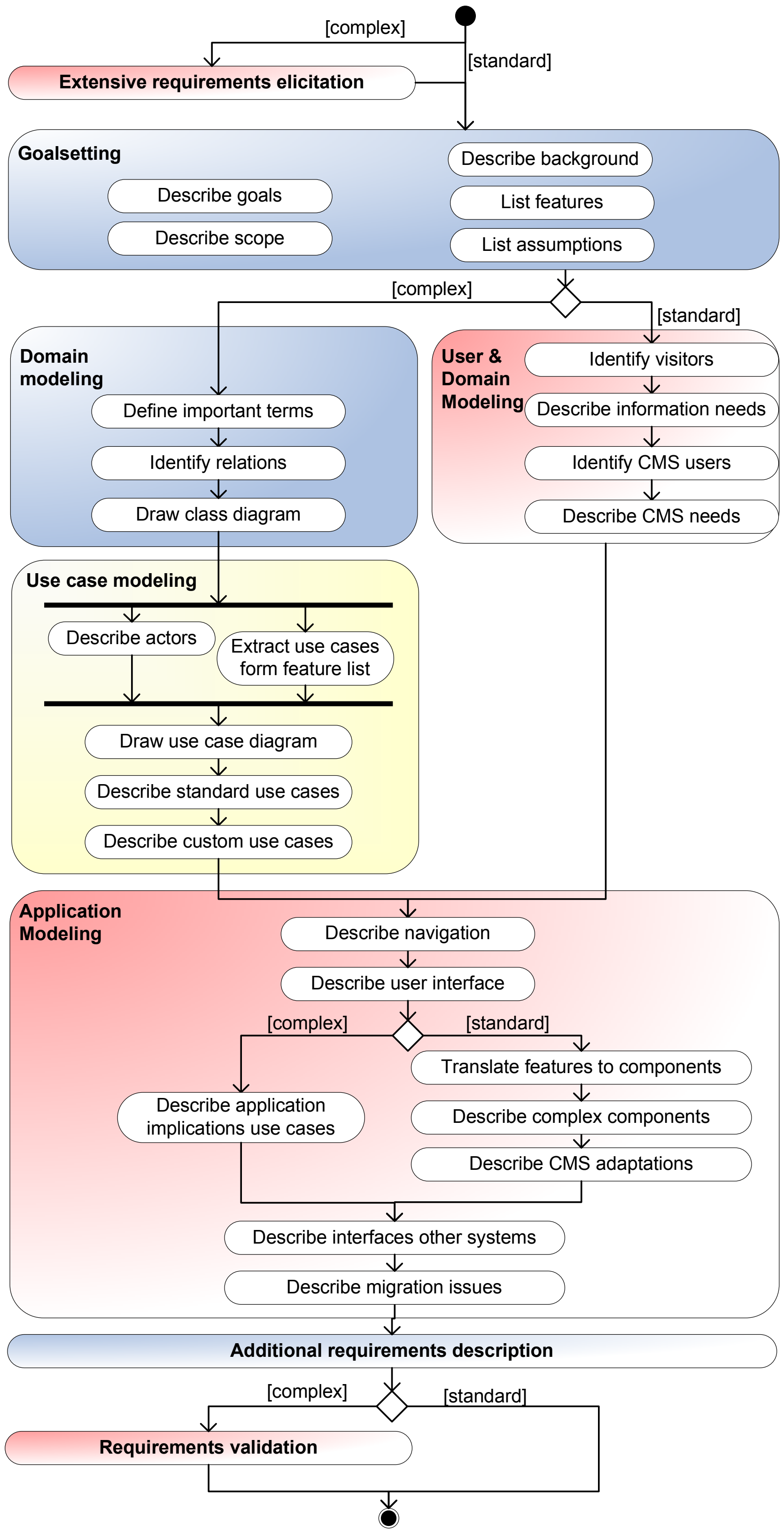


Figure 15-3: GX WebEngineering - Definition phase in WEM

16 Validation

Two types of validation are applied: expert validation and a case study. Because of scoping issues, only the definition phase was covered in this validation.

16.1. Expert validation

WEM was developed with input of the requirements management workgroup. The goal of this workgroup was an overall improvement in the requirements process at GX. Members of the workgroup were consultants and project managers of GX and one external consultant.

Two route maps of the method were validated, namely the standard and complex route maps. The method was assessed in this workgroup. The results were positive, for the following reasons:

- The distinction of standard and complex implementation situations was perceived as very useful.
- The use of user and domain modeling in the standard route was seen as very practical.
- Use case modeling in complex the complex route was seen as very useful.

Parts of WEM are already implemented in the organization. This is done by writing templates, organizing workshops and publishing WEM information in the intranet.

16.2. Case study

The route map for complex implementation situations was tested in a case study. In the case study the method was used in a project that consisted of building a Web application for a large telecommunication organization in the Netherlands. The purpose of the Web application was to support the testing of new products and services that are offered to a limited group of customers in a limited period of time. Employees of the organization should be able to develop and test a new offer with the application, without the help of GX. Several connections with existing back office systems had to be realized. Also, online payment of the products and services had to be supported. Finally, a special application for the Customer Care department needed to be developed, in order to support this department with customer service.

The project had an estimated budget of 400 man hours. Several stakeholders were involved in the requirements phase of the project. At the side of GX these were: a) project manager, b) consultants, and c) software architect. Stakeholders of the customer organization were: a) business project manager, b) technical project manager, c) Web department manager, and d) Customer Care project manager.

16.2.1. Usage of the method

Before the start of the project, a requirements document template was created. Also, a briefing was given to the consultants and project manager to outline the new method. The requirements analysis was carried out by GX consultants and reviewed by the project manager.

The requirements document consisted of thirty-two pages. The use case model consisted of seven actors, who were connected to seventeen use cases. Eight of these use cases were immediately translated to standard GX WebManager components. The others were more complex and were provided with use case descriptions. One part of the method was omitted, namely the drawing of a class diagram to model the domain, since the use was not necessary in this project.

In an interview, the consultants responded positive to the new method. In comparison to the old method, WEM was more structured and better able to describe complex functionalities. Also, the domain modeling was commented on as clarifying and useful. A remark was made on the use of a feature list, which was not recognized as very useful. Another comment was that use case modeling is a quite time-consuming task. However, the budgeted hours for this project were not exceeded.

16.2.2. Evaluation of the requirements document

The requirements document was send to the all stakeholders at GX and the customer. At GX, the requirements document was perceived as 'clear, structured, and with the right level of detail'. All stakeholders agreed that this document was an improvement to former requirements documents. However, the project manager expressed the fear that this method was too time-consuming.

To the project organization project members a survey was send (see Appendix I).The questions were divided in several categories. First, questions on the requirements process were asked in the categories (a) structure, (b) team, and (c) general. Then, questions on the requirements document were asked in the categories (a) understandability, (b) correctness, (c) use case modeling, and (d) general.

The answers to the survey appeared to be overwhelmingly positive. On a Likert scale of 1 to 5, where 1 was most negative and 5 most positive, a mean score of 4.4 was received. The given answer ranged from 3 to 5. No significant difference in scores was measured between the "process part" and "document part"

In summary, the requirements document was 'understandable and logical', with the right level of detail. Also, the functionalities described in the use cases perfectly matched the functionalities they wanted to be realized. Use cases were considered to be a great way to describe functionalities, since they are understandable for technical and non-technical project members.

16.2.3. Discussion

Summarizing, the results of the case study were positive. Nevertheless, some comments were made. First of all, only the definition phase was covered, which implies that the acquisition phase was done in the 'old-fashioned' way. The most obvious consequence was that the feature list, which should have been created in the acquisition phase and used in the definition phase, was seen as redundant by the consultants. The customer, however, did not comment on this.

Secondly, all project members at the customer's organization were familiar with use case modeling. If they were not, the requirements document might have been more difficult to understand.

Using the method may lead to new insights. The developed method is not static, but dynamic. Users of the method should adapt it to their own preferences. When appears that an activity structurally is omitted, the method should be updated.

17 Conclusions and discussion

17.1. Conclusions

In this chapter the main contributions resulting from this research project are described. The research question, described in the introduction, was:

> *"How should a design method be constructed for the process of developing web applications for GX WebManager?"*

This question is answered by describing an improved method engineering approach, and the result of this approach: the GX WebEngineering method.

In the next sections, a detailed description of the contributions resulting from this research is provided. First, the literature research is described; secondly, the contribution on the field of method engineering is outlined; and finally, the main deliverable, the GX WebEngineering method is described.

17.1.1. Literature study

First of all, the web content management world is relatively young. A lot of uncertainty existed on terminology. Therefore, a terminology list of the most used terms is presented. Also, the relationships between the different terms are described, which resulted in a schematic overview.

Secondly, in order to select candidate methods for the method base, a literature study to existing development approaches is conducted. This resulted in a schematic overview of methods, model and techniques, grouped by publishing year. Also the relations between the different approaches are indicated.

Thirdly, the methods Unified Process and UWE are described and analyzed. Process-data diagrams are provided to use them in a method base.

17.1.2. Method engineering

In this research project an improvement is proposed to the existing method engineering process. The described process helps in developing a method base, consisting of candidate methods that are selected based on how they meet the identified implementation situation needs.

Secondly, existing meta-modeling techniques appeared to be too complex to use. Therefore, an existing meta-modeling technique, resulting in a process-data diagram, is adopted and adjusted for the purpose of method engineering. By modeling the relations between activities and concepts, it is possible to engineer both process and data part of the method. Furthermore, with this meta-modeling technique this is done in a compact and elegant way.

Finally, an overview of characteristics of CMS-based web application implementations is provided. With these characteristics, one can categorize projects into standard or complex implementation situations.

17.1.3. GX WebEngineering Method

Looking at the delivered results of the research, a method has been developed and validated for developing CMS-based Web applications: the GX WebEngineering Method (WEM). The method comprises the acquisition, orientation and definition phases. WEM can be used for standard, complex and migration implementation situations, by following the described route maps. Since no such method existed, this research is an important addition to the existing information systems and Web development methods.

Also, the standard and complex route maps of the definition phase are validated and subsequently implemented at GX. This is done by translating the methods to templates and explanatory documentation, which are in turn published on the intranet.

17.2. Discussion and future research

Although the results are promising, several remarks should be mentioned.

First of all, a limitation on WEM is that only the acquisition, definition and orientation phases are covered. The purpose of this research was to develop also a new method for the design phase of a complex project, as has been done with the other phases. However, due to several practical constraints it was not possible to develop this new method. One example is that no literature has been found on the design phase of implementing CMS-based web applications. Apparently, this specialism is too young for this subject. It is recommended for future research on CMS-based web applications to give attention to the architecture modeling during an implementation. An opportunity for future research lies in extending WEM to the design, realization and implementation phases.

Secondly, only the definition phase has been validated. However, the results were promising. In the future, GX should expand and refine the method, based on experiences in executed projects. Also, besides the Unified Process and UML-based Web engineering, other relevant methods can be analyzed to improve the method base.

Thirdly, the meta-modeling technique that is used to model the method should be tested in other method engineering research projects. In the future, this technique should be updated, to keep it consistent with the UML standards.

Finally, research should be done in the integration of WEM in GX WebManager. As the content management system itself is capable to store structured documents, it makes sense to integrate the WEM design method as an extension of the WebManager product. This strategy is similar to the extension of the Oracle DBMS with Oracle CASE tools, or of the Baan ERP software with the Dynamic Enterprise Modeling (DEM) tooling (Brinkkemper, 1998).

References

Aydin, M.N. & Harmsen, F. (2002). Making a method work for a project situation in the context of CMM. *Lecture Notes in Computer Science, 2559*, 158-171.

Baresi, L., Garzotto, F. and Paolini, P. (2000) Extending UML for modeling web applications. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, HI, USA.

Baumeister, H., Koch, N., Mandel L. (1999). Towards a UML extension for hypermedia design. In R. France & B. Rumpe (Eds.*), Proceedings «UML» '99, LNCS, 1723*. Springer-Verlag, 614-629.

Berkum, M. van, Brinkkemper, S., Meyer, A. (2004). A Combined Runtime Environment and Web-Based Development Environment for Web Application Engineering. *CAiSE 2004*, 307-321

Bisbal, J., Lawless, D., Wu, B., Grimson, J. (1999). Legacy information system migration: A brief review of problems, solutions and research issues. *IEEE software, 16*, 103-111.

Booch, G. (1990). *Object oriented design with applications*. Redwood City, CA: Benjamin-Cummings Publishing Co., Inc.

Booch, G., Rumbaugh, J., Jacobson, I. (1999). *The unified modeling language user guide*. Redwood City, CA: Addison Wesley Longman Publishing Co., Inc.

Brinkkemper, S. (1996). Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, *38*(4), Elsevier Science Publishers, 275-280.

Browning, P., Lowndes, M. (2001), *JISC techwatch report: Content management systems* [technical Report TSW 01-02], Joint Information Systems Committee.

Burzagli, L., Billi, M., Gabbanini, F., Graziani, P. & Palchetti, E. (2004). The use of current content management systems for accessibility. *ICCHP 2004*, 331-338.

Chen, P. P. (1976). The entity-relationship model - Towards a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9-36.

Ceri, S., Fraternali, P., Bongio, A. (2000). A web modeling language (WebML): A modeling language for designing web sites. *Computer Networks, 33*, 137-157.

Conallen, J. (1999). Modeling web application architectures with UML. *Communications of the ACM, 42*(10), 63-70.

Conklin, J. (1987). Hypertext: An introduction and survey. *IEEE Computer Society Press, 20*(9), 17-41.

De Troyer, O. M. F. & Leune, C. J. (1998). WSDM: A user-centered design method for web sites. *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia.

Dietzsch, A. (2002). Adapting the UML to business modelling's needs - Experiences in situational method engineering. In: J.-M. Jézéquel, H. Hussmann, S. Cook (Eds.), *<<UML>> 2002 – Proceedings of the Unified Modeling Language;* Fifth International Conference, Dresden Germany, 73-83.

Falkenberg, E.D., Hesse, W., Lindgreen, P., Nilsson, B. E., Oei, J. L. H., Rolland, C., Stamper, R. K., Assche, F. J. M. V., Verrijn-Stuart, A. A., Voss, K. (1998). *A framework of information systems concepts – The FRISCO Report (Web edition).* [http://piano.dsi.uminho.pt/~jac/SI/zdocumentos/FRISCO.pdf, accessed 11/17/2004]

Fraternali, P. (1999). Tools and approaches for developing data-intensive web applications: a survey. *ACM Computing Surveys, 31*(3), 227-263.

Fraternali, P., Paolini, P. (1998). A conceptual model and a tool environment for developing more scalable and dynamic Web applications. In: H.-J. Schek et al. (Eds.), *Proceedings of the 6th International Conference on Extending Database Technology*, Valencia, Spain, 421-435.

Garzotto, F., Paolini, P., Schwabe, D. (1991). HDM - a model for the design of hypertext applications. *Proceedings of the 3rd annual ACM conference on Hypertext*, San Antonio, TX, 313-328.

Gellersen, H. W., and Gaedke, M. (1999). Object-oriented web application development. *IEEE Internet Computing, 3*(1), 60–68.

Gnaho, C. (2001). Web-based information systems development – A user centered engineering approach. *Lecture Notes in Computer Science, 2016*, 105-118.

Gray, E. M. and Tall, D. O. (1994). Duality, ambiguity and flexibility: a proceptual view of simple arithmetic, *The Journal for Research in Mathematics Education*, 26 (2), 115–141.

Halasz F., Schwartz M. (1990). The Dexter reference model. *Proceedings of the First Hypertext Standardization NIST Workshop*, Gaithersburg, MD.

Haniph, R. (2004). *Web Content Migration Project Design*, CMSwatch. [http://www.cmswatch.com/Features/TopicWatch/FeaturedTopic/?feature_id=105, accessed 04/28/2005]

Harmsen, F., Brinkkemper, S., Oei, J. L. H. (1994). Situational method engineering for informational system project approaches. *Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the IS Life Cycle*, 169-194.

Hirschheim, R., Klein, H.K. and Lyytinen, K. (1995). *Information systems development and data modeling; Conceptual and philosophical foundations*. England: Cambridge University Press.

IEEE (1998) Std 830-1998 Guide to Software Requirements Specifications (ANSI). *In Volume 4: Resource and Technique Standards.* The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection.

Isakowitz, T., Stohr, E. A., Balasubramanian, P. (1995). RMM: a methodology for structured hypermedia design. *Communications of the ACM, 38*(8), 34-44.

International Organization of Standardization. (2004, April*). ISO 9000: Frequently Asked Questions* [FAQ 029]. [http://www.iso.org/iso/en/iso9000-14000/iso9000/faqs.html, accessed 01/07/2005]

Jacobson, I., Booch, G., Rumbaugh, J. (1999). *The unified software development process*. Redwood City, CA: Addison Wesley Longman Publishing Co., Inc.

Jacobson, I., Christerson, M., Johnsson, P., and Overgaard, G. (1992). *Object-oriented software engineering: A use case driven approach*, Englewood Cliffs, NJ: Prentice-Hall.

Karlsson, F. (2002). *Bridging the gap between method for method configuration and situational method engineering*. Promote IT, Skövde, Sweden.

Karlsson, F., Ågerfalk, P.J. (2004). Method configuration: Adapting to situational characteristics while creating reusable assets. *Information and Software Technology, (46)9*, 619-633.

Koch, N. (1999). A comparative study of methods for hypermedia development. *Technical Report 9905*, Austria: Ludwig-Maximilians-Universität München.

Koch, N. (2001). *Software Engineering for Adaptive Hypermedia Applications.* Doctoral dissertation, Ludwig-Maximilians-Universität München, Austria: Uni-Druck Publishing.

Kruchten, P. B. (1995). The 4+1 view model of architecture. *IEEE Software, 12*(6), 42-50.

Kruchten, P. B. (1999). *Rational unified process — An introduction, (3rd ed.).* Redwood City, CA: Addison Wesley Longman Publishing Co., Inc.

Kumar K. & Welke R.J. (1992). Methodology engineering: A proposal for situation-specific Methodology construction. In: W. W. Cotterman & J. A. Seen (Eds.), *Challenges and Strategies for Research in Systems Development*. John Wiley & Sons Ltd.

Lang, M. (2001). Issues and challenges in the development of hypermedia information systems. *Proceedings of 11th Annual Business Information Technology Conference*, Manchester, England.

Lang, M. (2002). Hypermedia systems development: do we really need new methods? In E. Cohen & E. Boyd (Eds.), *Proceedings of the Informing Science + IT Education Conference*, Cork, Ireland, 883-891.

Lange D. (1996). An object-oriented design approach for developing hypermedia information systems. *Journal of Organizational Computing and Electronic Commerce, 6*(3), 269-293.

Lee H., Lee C. and Yoo C. (1998). A scenario-based object-oriented methodology for developing hypermedia information systems. In: R. Sprague (Eds.), *Proceedings of 31st Annual Conference on Systems Science.*

Lee, H., Kim, J., Kim, Y. G. Cho, S. H. (1999). A view-based hypermedia design methodology. *Journal of Database Management, 10*(2), 3-13.

McKeever, S. (2003). Understanding web content management systems: Evolution, lifecycle and market. *Industrial Management and Data Systems*, *103*(9), 686-92.

Object Management Group (2004). *UML 2.0 superstructure specification.* Technical Report ptc/04-10-02.

Ralyté, J,, Deneckère, R. and Rolland, C. (2003). Towards a generic model for situational method engineering. *Lecture Notes in Computer Science, Vol. 2681*, Springer-Verlag, 95.

Rational Software Corporation (1999). *Rational unified process.* Version 5.1 (build 12).

Rossi, M., Brinkkemper, J. (1996). Complexity metrics for systems development methods and techniques. *Information Systems, 21*(2), 209-227.

Rossi M., Tolvanen J.-P., Ramesh B., Lyytinen K., Kaipala J. (2000). Method rationale in method engineering. *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS-33),* IEEE Computer Society Press.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorenson, W. (1991). *Object-oriented modeling and design*. Englewood Cliffs, NJ: Prentice Hall.

Saeki M. (2003). Embedding metrics into information systems development methods: An application of method engineering technique. *CAiSE 2003,* 374-389.

Sauer, S., Engels, G. (1999). Extending UML for modeling of multimedia applications. In: M. Hirakawa, P. Mussio (Eds.), *Proceedings of the IEEE Symposium on Visual Languages*, Tokyo, 80-87.

Schwabe, D., de Almeia Pontes, R. (1998). OOHDM-WEB: Rapid prototyping of hypermedia applications in the WWW, *MCC 08/98*, Brasilia: Department of Informatics, Pontifícia Universidade Católica do Rio de Janeiro.

Schwabe, D., Rossi, G. (1995). The object-oriented hypermedia design model. *Communications of the ACM, 38*(8), 45-48.

Slooten, K. van & Brinkkemper, S. (1993). A method engineering approach to information systems development. *In Information Systems Development Process*. N. Praksh, C. Rolland, B. Pernici (Eds.). Elsevier Science Publisher B.V. (North- Holland). 167-186.

Slooten, K. van & Hodes, B. (1996). Characterizing IS development projects. *In: Method Engineering: Principles of method construction and tool support, Proceedings of the IFIP TC8, WG8.7/8.2 Working conference on method engineering*, Atlanta, USA.

Souer, J., Van de Weerd, I., Versendaal, J., Brinkkemper, S. (2005). Developing situational content management system-based web applications. *WISE 2005.*

Vaishnavi, V. and Kuechler, W. (2004). *Design Research in Information Systems.* [http://www.isworld.org/Researchdesign/drisISworld.htm, accessed 01/28/2005]

Vidgen, R., Goodwin, S. & Barnes, S. (2001). Web Content Management. *Proceedings of the 14th International Electronic Commerce Conference*, Bled, Slovenia, 465-480.

Weerd, I. van de, Souer, J., Versendaal, J., Brinkkemper, S. (2005). Situational web content management implementations. *Proceedings of the 1st international workshop on situational requirements engineering processes*, Paris, 13-30.

Appendix 1: Case study survey

Achtergrond
Dit onderzoek vindt plaats binnen mijn afstudeerproject "Design Methods for WebManager Implementations" van de opleiding Business Informatics, Universiteit Utrecht. In het kader van dit project richt ik mij op het optimaliseren van de de requirements analyse methode die GX gebruikt.

Enquête
Dit is een enquête over het requirements traject van het uitgevoerde project. Eerst worden een aantal vragen over het requirements *proces* gesteld en vervolgens een aantal vragen over het requirements *document*. Wanneer u weinig of niet bij het proces aanwezig bent geweest, kunt u het eerste gedeelte overslaan en alleen het tweede gedeelte invullen.

De vragen kunnen beantwoord worden door één van de radio buttons aan te klikken. De schaal loopt van -- (zeer slecht, of zeer mee oneens), via 0 (voor neutraal), tot ++ (zeer goed, of zeer mee eens). Voor beiden onderdelen is er ook de mogelijkheid om extra opmerkingen in het tekstveld te schrijven.

Hartelijk dank voor uw medewerking.

Requirements Proces
Hieronder volgen een aantal vragen over het requirements proces.

1. Structurering

Is het requirements proces voldoende gestructureerd verlopen? Het requirements proces omvat aspecten zoals communicatie, afspraken en planning.				
○ --	○ -	○ 0	○ +	○ ++

Zijn de requirements sessies voldoende gestructureerd verlopen?				
○ --	○ -	○ 0	○ +	○ ++

Heeft GX tijdens de requirements sessies voldoende aangegeven wanneer van de scope afgeweken is?				
○ --	○ -	○ 0	○ +	○ ++

2. Team

Heeft u naar uw mening met de juiste mensen aan tafel gezeten?				
○ --	○ -	○ 0	○ +	○ ++

Hebben de mensen van GX goede sturing aan het traject gegeven?				
○ --	○ -	○ 0	○ +	○ ++

3. Algemeen

Hoe beoordeelt u het totale requirements proces?				
○ --	○ -	○ 0	○ +	○ ++

Opmerkingen requirements proces:

Requirements Document
Hieronder volgen een aantal vragen over het requirements document.

1. Begrijpbaarheid

Vindt u de structuur van het requirements document logisch?				
○ --	○ -	○ 0	○ +	○ ++

Zijn de functionaliteiten concreet genoeg beschreven?				
○ --	○ -	○ 0	○ +	○ ++

Is de mate van detaillering voldoende?				
○ --	○ -	○ 0	○ +	○ ++

Is de onderlinge samenhang tussen de verschillende hoofdstukken duidelijk?				
○ --	○ -	○ 0	○ +	○ ++

2. Correctheid

Worden met dit systeem de 'business doelen' gerealiseerd die u voor ogen heeft?				
○ --	○ -	○ 0	○ +	○ ++

Komen de functionaliteiten die benoemd zijn overeen met de functionaliteiten die u gerealiseerd wilt zien?				
○ --	○ -	○ 0	○ +	○ ++

3. Gebruikte methode

Was u bekend met het gebruik van use cases om functionaliteiten te beschrijven?				
○ --	○ -	○ 0	○ +	○ ++

Wat vindt u van het gebruik van use cases om de functionaliteiten te beschrijven?				
○ --	○ -	○ 0	○ +	○ ++

Is de relatie tussen de use cases (in het hoofdstuk Use Case Model) en de uitwerking (in het hoofdstuk Applicatie Inrichting) duidelijk?				
○ --	○ -	○ 0	○ +	○ ++

4. Algemeen

Hoe beoordeelt u het requirements document?				
○ --	○ -	○ 0	○ +	○ ++

Opmerkingen requirements document:

Naam:

E-mail:

Submit