

Requirements Engineering in the Rational Unified Process

Johannes Passing

Hasso-Plattner-Institute for Software Engineering, D-14482 Potsdam
johannes.passing@hpi.uni-potsdam.de

Abstract. This paper aims at providing an overview of Requirements Engineering in the context of the Rational Unified Process. After a brief introduction to the general concepts of the Rational Unified Process, the requirements-related Disciplines “Business Modeling” and “Requirements” are described in more detail.

1 Introduction

Capturing and managing requirements is a key factor to the success of a software development effort. As such, it is both obvious and reasonable to tightly integrate the requirements-related activities with the rest of the software engineering process. The Rational Unified Process (RUP) [3], targeted at covering the complete development lifecycle of a software project, implements this integration by incorporating Requirements Engineering as one of its core disciplines.

One of the main difficulties of requirements capture is the fact that stakeholders alone, once identified, can neither be expected to specify all requirements at the beginning of the project nor to specify each requirement of the system to be developed in the desired precision. Whilst this observation can hardly be influenced by the practices deployed, the method of capturing and maintaining these requirements also has great influence on the success of the project.

Classic requirements capture, as criticized by RUP, mainly focuses on “what” is to be developed, and partly “how” it should be developed, while often neglecting the individual intents and “why”s behind each requirement [1].

Regarding a common sentence like “If event x occurs, the user should receive an email notification”, it may not be obvious why the email notification is valuable in this situation and consequently, it may be hard to decide on the importance of this specific requirement. In contrast, the better analysts and developers understand the context and the rationale of each requirement, the easier it is to comprehend and interpret the requirement itself and consequently, to develop a suitable solution.

Another criticism stated by Kruchten [1] concerns the Waterfall Process [4]. In the Waterfall Process, requirements are largely assumed to be captureable completely at the beginning of the project. As such, requirements are assumed to be stable throughout the rest of the project and can thus serve as a reliable foundation for further planning. This, however, neglects the fact that not only

user requirements are likely to change at a later stage, but also that technological requirements should not be assumed to be stable.

2 The Rational Unified Process

The Rational Unified Process is an iterative software engineering process consisting of both a process framework and a process product. As a process product, it consists of a comprehensive HTML-based documentation [6, 7], including support material such as checklists or document templates. Initially released by the Rational Software Corporation, RUP is now maintained and distributed by IBM as a commercial, non royalty-free product [6]. As a process framework, it provides an approach to organizing and assigning tasks within an organization. The six main practices covered by RUP are [1]:

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

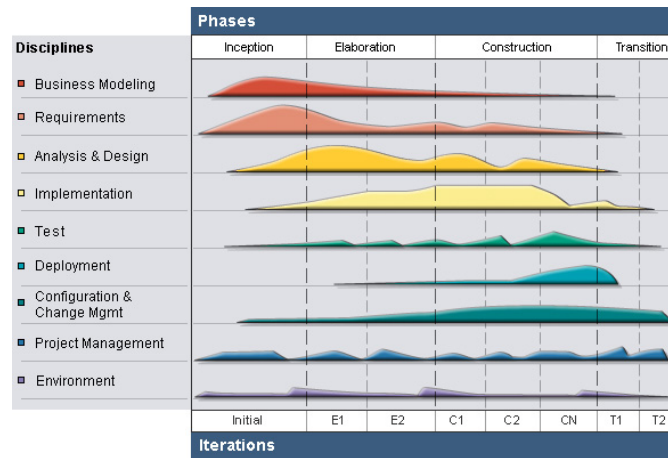


Fig. 1. Phases and Disciplines in the RUP ([6])

The process described by RUP can be decomposed into Phases and Disciplines, the latter also referred to as workflows¹. While Phases represent the temporal aspect of the development process, Disciplines are logical groupings

¹ The term “Workflow” has been replaced by “Discipline” in RUP version 2002

of activities. Capturing and maintaining requirements is covered by the Requirements Discipline and, as noted above, is thus fully incorporated into this structure. As can be seen in Figure 1, the Disciplines Requirements and Business Modeling, which will be further explored in the following sections, are not limited to the Inception Phase. Instead, both are part of all Phases, emphasizing the fact that capturing and maintaining requirements remains of importance during the complete development lifecycle.

2.1 Structure

Both the process framework and the documentation rely on a certain structure, Phase and Discipline being two of the core elements.

Disciplines can be further decomposed into Activities, which denote a coarse grained area of activity that is to be addressed within the Discipline. Activities in turn refer to a set of Tasks, whereas a Task can be part of several Activities. A Task describes a certain unit of work that yields a meaningful result. For documentation purposes, RUP also includes a step-by-step description for each of these Tasks. Common to all Tasks is the property that they consume certain Artifacts and, as the work result, yield new or modified Artifacts.

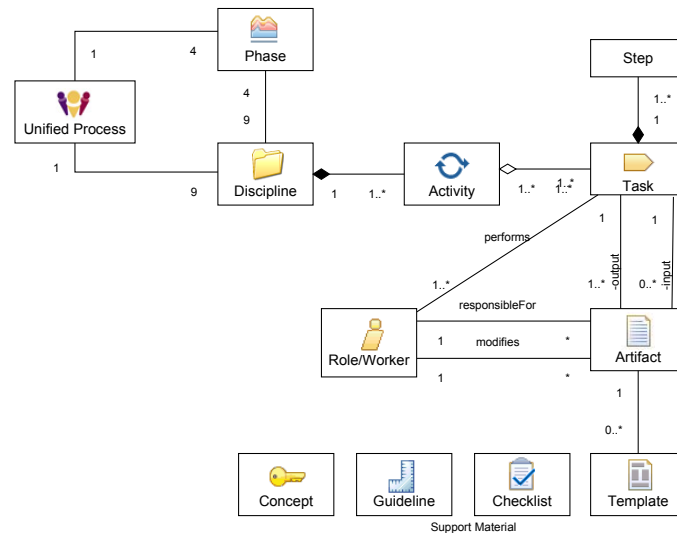


Fig. 2. Structure of RUP

In the context of RUP, the notion of an Artifact is not limited to documents. For example, also models and requirements are referred to as Artifacts. Each of the Artifacts falls under the responsibility of a certain Role.

A role, also referred to as Worker in [2], defines a certain set of skills, competences and responsibilities, that can be realized by an individual internal to the

organization. There may be several employees realizing a specific role as well as an employee acting in several roles.

2.2 Tailoring

Especially for smaller projects, addressing all Disciplines, activities and tasks may impose too much overhead on the project. In order to avoid bureaucracy and still enable smaller projects to benefit from the concepts and best practices employed in RUP, the process may be customized to suit the individual needs. This customization, referred to as Tailoring, can also be used to combine RUP with practices of agile processes such as Extreme Programming [12] or Scrum [11].

2.3 Overview of Requirements Capture

Three main aspects of requirements capture can be identified in RUP:

1. Understand System Context
2. Capture functional requirements
3. Capture non-functional requirements

To be able to effectively and successfully capture and maintain requirements, a certain understanding of the context of the system to be developed is necessary. Given the example of a transaction processing system for a financial institute that is to be implemented by an external company, it is a prerequisite for the implementer to acquire a certain domain knowledge. Capturing and maintaining this context information is addressed by the Business Modeling Discipline, which will be further explored in the next section.

Collecting, refining and maintaining both functional and non-functional requirements is the objective of the second Discipline in focus of this paper, the Requirements Discipline.

3 Business Modeling Discipline

Complex software systems are often not built by organizations themselves, but instead by external service providers [2]. As a consequence, these service providers do not have inherent knowledge of the environment the software system is developed for. Besides establishing a basic knowledge of the problem domain, it is thus essential for the developing organization to investigate this environment. Knowing and understanding this context then enables the service provider to identify and understand the problems and, coming along with that, to identify improvement potentials.

Aim of the Business Modeling Discipline is to provide a structured approach to identify structure and dynamics of the target organization as well as to establish a common, conflict-free understanding among the customers, end users and developers. To achieve this, Business Modeling employs modeling techniques

similar to those for software engineering, which will be presented in the following sections.

Unsurprisingly, the importance of the Business Modeling Discipline is directly related to the complexity and size of the project. Whilst it may prove valuable in larger projects, it may be unnecessary overhead for smaller projects or projects, where this common understanding has already been established. As such, this Discipline is regarded optional and, for example, is also not contained in the “small configuration template”, offered by IBM, which denotes a tailored version of RUP.

3.1 Domain Modeling

One approach to investigate the context of the system is Domain Modeling. This technique focuses on the things that exist or play a certain role in the context of the system. This includes, but is not limited to, business objects, concepts or certain events.

Common practice in Domain Modeling is therefore to have both modeling specialists and domain experts to attend workshops, in which the necessary information will be gathered and modeled appropriately. The concepts and domain entities can then be incorporated and structured in a UML [10] class diagram, showing the relationships among the individual elements. These diagrams are typically of modest size (10 to 50 classes) and may be used as a medium to communicate the domain concepts and entities. They are, however, not suitable or intended for being able to serve as a foundation for a later implementation.

Along with creating the class diagrams, a glossary may be created, containing a short explanation of each of the concepts and terms involved. In less complex environments projects, the class diagram may also be omitted in favor of a glossary.

3.2 Business Modeling

In larger or more complex situations, domain models alone may not be enough to explain and communicate the context. In this case, Business Modeling, a superset of Domain Modeling may be more applicable.

As in the case of Domain Modeling, identifying domain classes, in this case named Business Entities is one of the aims of this approach. Additionally, Business Actors, Workers as well as Business Use Cases are being modeled.

Business users such as customers or partners are referred to as Business Actors. A Business Worker entitles a role people play within a company. An example for a Business Worker may thus be the clerk in the branch office of a bank. Business Use Cases describe processes performed within the organization

By displaying the processes rather than only the “things”, Business Modeling establishes a broader, more thorough view of the organization than Domain Modeling. This broader view and the focus on use cases may especially be valuable when implementing systems that have a larger impact on an organization.

Rather than adopting existing processes, several processes may in this case be altered, dropped or introduced - as such, reflection on how the business is run [1] is an important part of this modeling process.

In the case of Domain Modeling, the value and conciseness of the class diagrams created are largely dependent on the competence and experience of the individual domain experts. By applying a more structured procedure, Business Modeling tries to reduce this dependency. Moreover, as Business Workers, Entities and Use Cases are embedded into more comprehensive models, the rationale behind each of these elements becomes more clear and can be traced back to the customer more easily.

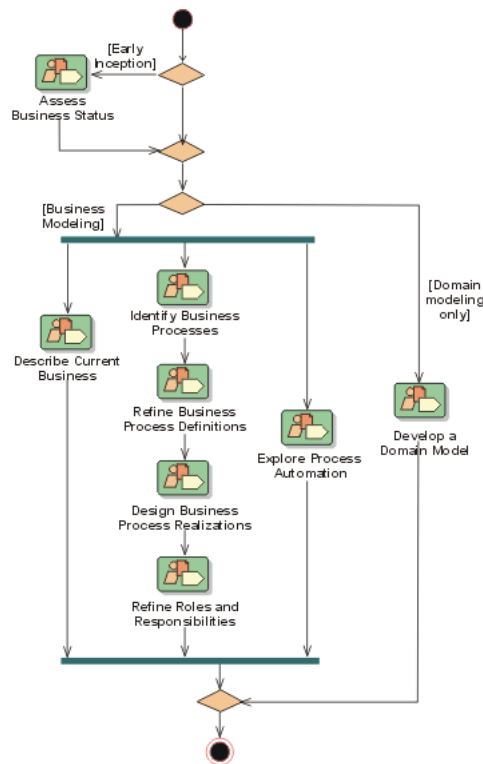


Fig. 3. Business Modeling Discipline ([7])

Figure 3 shows the sequence of activities performed in Business Modeling. Domain Modeling is shown as an Activity on the right, which nicely illustrates the relationship between these two approaches.

Like all other Activities and Disciplines in RUP, Business Modeling involves dealing with several Artifacts. The most important among these, especially in re-

gard of the relationship to the Requirements Discipline, are the Business Object Model and the Business Use Case Model.

Business Use Case A Business Use Case identifies a service or value offered by the organization to an external stakeholder (Business Actor), such as a customer or partner. As such, it denotes the most abstract view of the company, i.e. only the interactions with external entities are shown while the internal processes of the organization are hidden.

Given the example of a financial institute, the following simple business use case is conceivable. It shall be noted that a special UML Profile [5, 8] is deployed for Business Use Case modeling, which also makes the models distinguishable from (System) Use Cases.

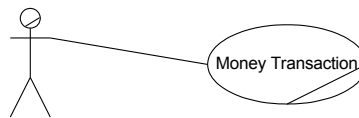


Fig. 4. Sample Business Use Case Model (derived from [1])

Business Object Model The Business Object model describes realizations of Business Use Cases in the form of an object model, thus representing a more inside view of the business. Following the example of the financial institute, it may now be modeled that the “Money Transaction” Use Case is indeed realized by two workers, one of them a case worker interacting with the customer, the other one being an internal worker only. As resources, they have access to the customer profile, the account an loan data.

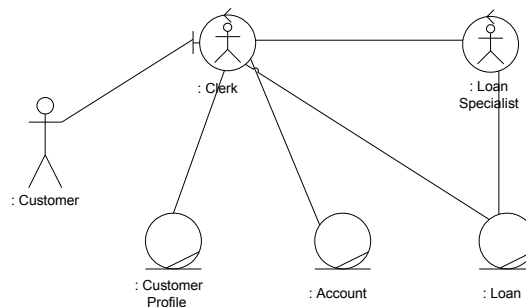


Fig. 5. Sample Business Object Model (derived from [1])

As in the case of Business Models, special symbols are used for these Business Object Models [5].

From the Business Model to the systems While creating Business Use Case and Object Models helps structuring and maintaining information about the organization, these models also provide a starting point for deriving System Use Case Models, which are dealt with in the Requirements Discipline. System Use Cases, in contrast to Business Use Cases, denote the more classic usage of use cases, i.e. they describe the use cases implemented by a software system as well as the Actors interacting with the system.

The first question that is to be posed for each individual Business Worker mentioned in the Business Object Model is whether or not the worker will interact with the system to be built. Consequently, each interacting worker will then become an Actor in the System Use Case Model. Furthermore, for each Business Use Case the Business Worker participates in, a System Use Case can be identified.

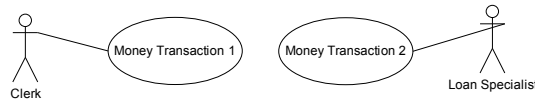


Fig. 6. Derived System Use Case Model (derived from [1])

Performing these steps for all Business Workers yields an initial System Use Case Model, as illustrated by Figure 6. As the Customer, represented by a Business Actor in the Business Use Case and Object Model, is both external to the company and has no direct interaction with the system, he/she is not represented in the initial System Use Case Model.

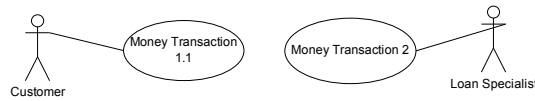


Fig. 7. Derived System Use Case Model (derived from [1])

Starting from this model, one may reflect on automating certain Business Workers. Following the example of a transaction processing system to be implemented for a bank institute, it would be well conceivable to let the customer directly interact with the system instead of communicating with the Clerk as a mediator. Figure 7, showing a refined System Use Case Model, incorporates this idea - the Actor “Clerk”, who had been derived from the corresponding Business

Worker has now been replaced by the customer, who, for example, now interacts with our system by using a web interface or similar facilities. Considering the fact that we have essentially replaced a Business Worker by a Business Actor, it now becomes obvious that we have changed the way the business is performed [7] - responsibilities of the Business Worker, an employee, have been transferred to the Business Actor, a customer.

Though Business Modeling or Domain Modeling may be a too time-consuming process for smaller projects, it provides valuable information to the other Disciplines. Business Use Case Model, Business Object Model as well as Domain Model are well-suited to describe the concepts and processes of the organization in scope of the project. Also, it may be assumed that by creating the model and by capturing the information required to develop these models, the analysts gain a much deeper insight into the processes of the organization and the rationales behind the system to be developed than by only focusing on the requirements formulated by the stakeholders.

4 Requirements Discipline

Aim of the Requirements Discipline is to capture, structure and maintain requirements formulated by stakeholders as well as to manage changing requirements. As most of the activities performed by other Disciplines are directly or indirectly dependent on requirements, the Requirements Discipline has an important position in the overall development process.

Besides providing a significant portion of the input to other Disciplines such as Analysis and Design, solid requirements management is also a prerequisite for being able to estimate cost and effort the development project will involve.

Especially in larger projects, the Requirements Discipline can significantly profit from Business Modeling. The Business Modeling Discipline provides valuable information about the organization and the context of the system, but, as described before, can also yield some of the initial use cases. The decision whether to perform or to omit the Business Modeling Discipline therefore influences the way the Requirements Discipline is performed.

The actual process employed by the Requirements Discipline is largely focused on Use Cases, which are assumed to be an appropriate medium for communicating functional requirements among stakeholders and developers. For this purpose, Use Cases as well as all other specifications and documents dealt with in this Discipline should also be authored in the customers language.

Figure 8 shows the workflow used in the Requirements Discipline. Unlike the workflow of the Business Modeling, the flow is not sequential but contains several cycles. This, as well as the fact that the complete activity diagram is executed repeatedly, emphasizes the fact that RUP is an iterative process.

In the following sections, each activity involved in the Requirements Discipline is briefly explained. Moreover, as each activity may introduce additional tasks and artifacts, several diagrams showing the relationships among tasks and artifacts will be shown.

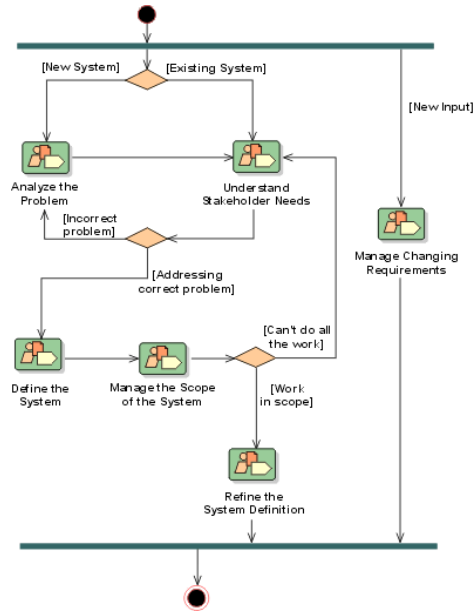


Fig. 8. Requirements Discipline ([7])

4.1 Activity: Analyze the Problem

The activity “Analyze the Problem” describes the initial tasks to be performed if a new system is to be developed. The most essential aspect of this activity is to identify the key stakeholders as well as the key requirements for the project.

In order to sketch out the boundaries of the system, part of the Develop Vision Task is to achieve agreement on a the actual problems to be solved. Based on this information, the first Artifact, the Vision Document, may be authored. This document is intended to describe the overall vision of the project as well as to document the information about the key stakeholders and problems identified so far.

To avoid misunderstandings among the parties involved in the project and to ease the process of establishing a common understanding and vision, the task “Capture common vocabulary” collects the most important terminologies used in the problem domain. This information is then gathered and structured in the Glossary, which is maintained and enhanced throughout the rest of the development process.

4.2 Activity: Understand Stakeholder Needs

Having identified the stakeholders in the previous activity, more detailed requirements may now be collected. The task “Elicit Stakeholder Requests” embraces the process of interviewing customers, evaluating questionnaires and other techniques aimed at gathering more specific requirements. One of the techniques

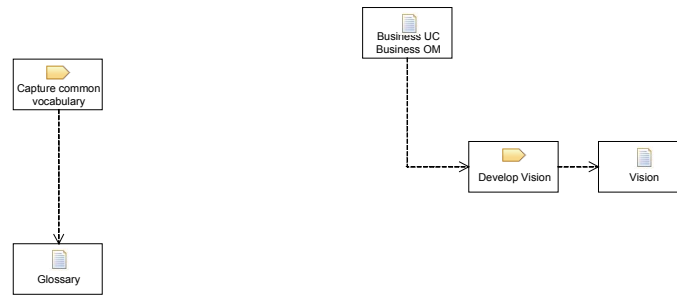


Fig. 9. Tasks and Artifacts involved in Activity: Analyze the Problem

highlighted by RUP is the application of Storyboarding, in which Story Boards, each describing a specific scenario or behavior of the system, are created and refined by users.

In order to be able to comprehend the intentions behind the individual requirements formulated at this and later stages, it will now be helpful to have established a solid understanding of the context of the system, as addressed by Business Modeling.

In parallel to gathering functional requirements, non-functional requirements should now be identified. In contrast to functional requirements, which will mainly be incorporated in Use Cases, non-functional requirements are gathered and structured in a separate Artifact, the Supplementary Specification.

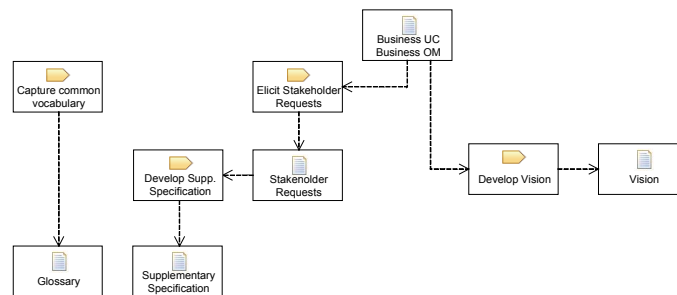


Fig. 10. Tasks and Artifacts involved in Activity: Understand Stakeholder Needs

4.3 Activity: Define the System

After the vision has been defined and the key stakeholders and requirements have been identified by the previous two activities, this information can be analyzed

further. The most important Task involved in this Activity is thus “Find actors and Use Cases”, which, as the name suggests, aims at identifying and further refining System Actors and Use Cases based on the information gathered.

If Business Modeling is being performed in parallel, both initial Business Use Case Models and Object Models are now available and, as described in the previous example, may be used to identify these Actors and to derive Use Cases.

The Use Cases and Actors identified should then be further explained. Though the precise definition is elaborated in later activities, each Use Case and Actor should be accompanied by a brief, customer-understandable description. In the case of Actors, individual responsibilities should be described as well as the value the system provides for this specific Actor. In the case of Use Cases, an overall description as well as a coarse flow of events ought to be specified.

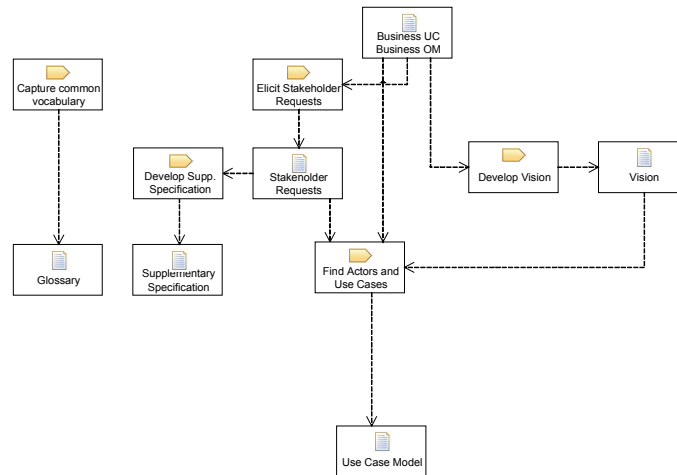


Fig. 11. Tasks and Artifacts involved in Activity: Define the System

4.4 Activity: Manage the Scope of the System

Resulting from the previous activities, a significant number of Use Cases may now have been identified. The focus of the activity is to structure the Use Case Models and to prioritize individual Use Cases.

This prioritization is addressed by the Task “Prioritize Use Cases”, which relies on several criteria to perform the prioritization. Besides taking into account the value of each Use Case provided to stakeholders as well as the budget and time provided, this Task uses two Artifacts from other Disciplines being performed in parallel.

The “Project Management” Discipline provides the “Risk List”, an Artifact documenting events that could lead to a negative outcome of the project. Based

on this information, those Use Cases that impose a larger risk on the success of the development can be prioritized to be implemented early while low-risk Use Cases may be deferred to a later stage.

The second artifact applied is the “Software Architecture Document”, created by the “Analysis and Design” Discipline. This document describes an initial draft of a software architecture facilitating the implementation of the system. The fact that the chosen architecture may imply certain Use Cases to have a larger impact on the architecture than others should be taken into consideration when prioritizing the individual Use Cases.

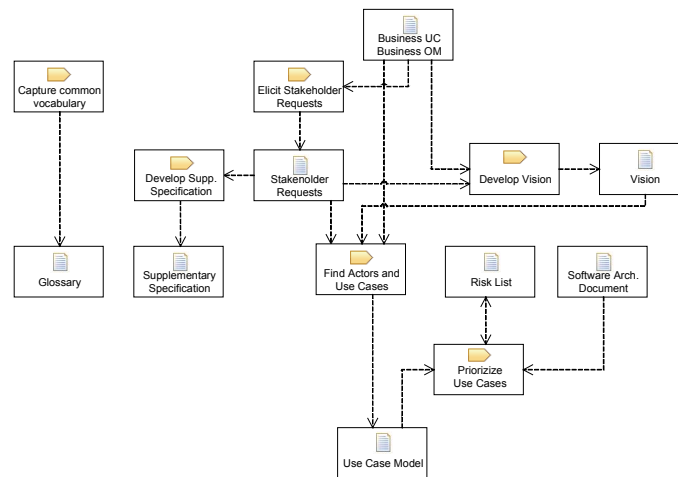


Fig. 12. Tasks and Artifacts involved in Activity: Manage the Scope of the System

4.5 Activity: Refine the System Definition

At this stage, most requirements have been captured and the first Use Cases and specifications have been created. Part of the “Refine the System Definition” Activity is now to foster the understanding of the project scope reflected by the prioritized feature set yielded by the previous activity. On the other hand, the Activity includes detailing existing Artifacts and is hence aimed at completing the first stage of Requirements Engineering.

Each of the Use Cases defined in the “Define the System” and other Activities, are now further elaborated. This may happen by providing detailed textual descriptions as well as by creating state charts or activity diagrams addressing the flow of events in each Use Case in detail. Especially in the case of user interfaces, prototypes may be created to gather feedback and further proposals from users and stakeholders.

Similar to functional requirements expressed by Use Cases, non-functional requirements contained in the Supplementary Specification should be further specified and completed.

Optionally, a Software Requirements Specification (SRS) may be created in this stage of the process. The SRS depicts a single document or a collection of Artifacts consolidating and describing the complete set of requirements captured and defined to this stage. The SRS may be authored in a text-only form as well as a combination of text and use case models [9]. The yielded documents are, however, not final but are subject to updates throughout the project lifecycle in order to continuously reflect the most current set of requirements.

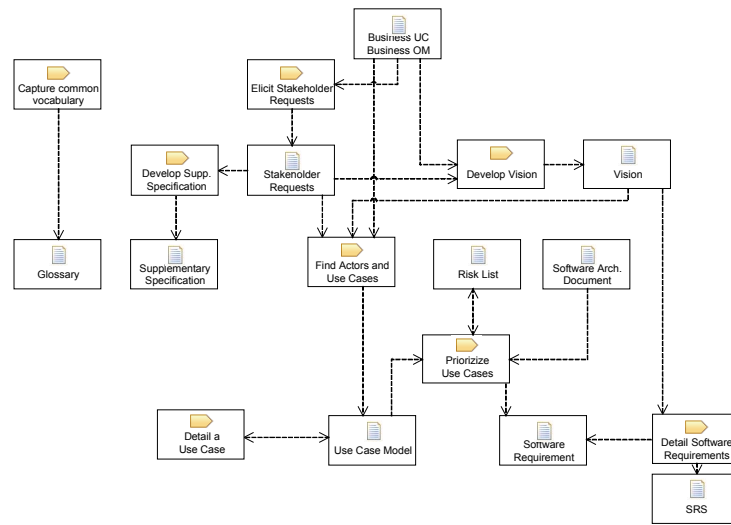


Fig. 13. Tasks and Artifacts involved in Activity: Refine the System Definition

4.6 Activity: Manage changing requirements

As stressed before, RUP emphasizes the fact that requirements are likely to change throughout the development lifecycle. Managing these changing requirements is the aim of the Activity “Manage changing requirements”.

One aspect of this Activity is thus to assess the impact of changing requirements on other requirements and to incorporate them into the system being developed.

Another aspect of this Activity is to enhance and maintain the various models created, which is addressed by the Task “Structure Use Case Model”. This includes reorganizing models as well as to find generalizations and include-relationships among Use Cases. As such, this Task helps both maintaining and streamlining these Artifacts.

Moreover, regular meetings with stakeholders should be held in which requirements are reviewed. Besides ensuring that the right product is being developed, especially the requirements to be implemented in the next period of time should be carefully reviewed.

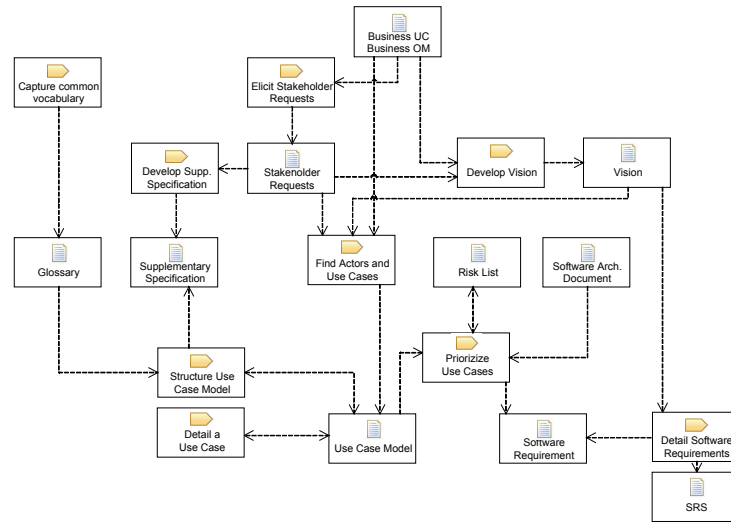


Fig. 14. Tasks and Artifacts involved in Activity: Manage changing requirements

5 Concluding Remarks

While the Rational Unified Process as well as its Requirements Engineering components are too complex to be covered in depth by this paper, an overview of the activities and artifacts involved in the process of capturing, refining and maintaining requirements should have been presented.

The most significant drawback of the Rational Unified Process as a whole might be its complexity. In respect thereof, RUP requires each developer or architect involved in the process to know the basic concepts and ideas and gain an overview of the process. This burden, however, may be lowered by the ability of RUP to be tailored to the individual needs. Furthermore, the interactively usable documentation and the support material provided by RUP are well suited for being integrated into the daily work of developers, architects and analysts.

Requirements Engineering, as addressed by the Business Modeling and Requirements Disciplines is fully incorporated into the RUP development process. The Use Case centered approach employed by these Disciplines depicts a structured method, yielding artifacts that represent a viable compromise between being technically valuable and at the same time being well-suited for communication with stakeholders.

6 Summary

Following a discussion of the general ideas and the structure of the Rational Unified Process, the two Disciplines related to Requirements Engineering, Business Modeling and Requirements, have been presented. Finally, the individual activities involved in the Requirements Discipline have been portrayed.

References

1. P. Kruchten, The Rational Unified Process, Addison Wesley, 2001
2. I. Jacobsen, G. Booch, J. Rumbaugh, The Unified Development Process, Addison Wesley, 2001
3. IBM Corporation, Rational Unified Process,
<http://www-306.ibm.com/software/awdtools/rup/> (07/21/2006)
4. Parametric Cost Estimating Handbook, The Waterfall Model,
<http://www1.jsc.nasa.gov/bu2/PCEHHTML/pceh.htm> (07/12/2006)
5. IBM Corporation, Rational UML Profile for Business Modeling,
http://www.cs.joensuu.fi/pages/ageenko/teaching/OOD/RUML_Business_Modeling.pdf
(06/02/2006)
6. IBM Corporation, Rational Unified Process (Evaluation Version), Version 7.0
7. Rational Software Corporation, Rational Unified Process, Version 2002.05.00
<http://www.ts.mah.se/RUP/RationalUnifiedProcess/> (06/02/2006)
8. Object Management Group, Uml Extensions,
http://www.jeckle.de/files/UML1.2/4_exten.pdf (06/02/2006)
9. L. Probasco, D. Leffingwell, Combining Software Requirements Specifications with Use-Case Modeling, Rational Software,
http://www.spc.ca/downloads/srs_usecase.doc (05/10/2006)
10. Object Management Group, Unified Modeling Language (UML), Version 2.0,
<http://www.omg.org/technology/documents/formal/uml.htm> (2006/07/20)
11. Scrum, Control Chaos, <http://www.controlchaos.com/> (2006/07/20)
12. K. Beck, Extreme Programming Explained. Embrace Change, 2nd Edition, Addison Wesley, 2004