Experimenting a Requirements Engineering Process Based on Rational Unified Process (RUP) Reaching Capability Maturity Model Integration (CMMI) Maturity Level 3 and Considering the Use of Agile Methods Practices

Caroline Carbonell Cintra B.Sc.
Instituto de Informática
Universidade Federal do Rio Grande do Sul
DBServer Assessoria em Sistemas de Informação
cccintra@inf.ufrgs.br

Roberto Tom Price Eng. M.Sc. D.Phil.
Instituto de Informática
Universidade Federal do Rio Grande do Sul
tomprice@terra.com.br

Abstract

This work has the purpose of describing a software development process with the following characteristics: its scope lies within the requirements engineering activities; it fulfills CMMI requisites for Requirements Management and Requirements Development process areas (maturity level 2 and 3, respectively); it is based on RUP practices and activities where possible practices of agile methods are employed.

Related work is considered and similarities as well as differences to the process proposed here are pointed out. Such process is presented in terms of each of its activity flows, including mention to the artifacts and roles involved on the activities. CMMI fulfillment is also described, followed by the description of the main contributions achieved by this work and comments on future development.

1. Introduction

The current spectrum of software development processes has been characterized by a constant revaluation of the methodologies used in each organization [3], by the value perceived in quality and productivity as means to increase return on investment [22] and by the pursue of continuous improvement [23] as a way to increase the usually low rates of success and satisfaction achieved by software projects [10].

The catalogue of development approaches available to researchers and practitioners offers a variety of methods and techniques as well as enables adaptation of these approaches, according to organization type, project size, requirements stability, etc. These approaches may be described as specifications or as process models [30], [24], [16], [17], may represent a development framework, composing a generic process that needs to be instantiated in each project [20], [11], or may be described by a set of fundamental orientations based on principles and practices to be followed [5].

In the matter of software development approaches, special interest has been dedicated to *requirements engineering*. Requirements are the starting point of all software system definition and therefore they are crucial factors for the success of any software project final product. Requirements engineering is pointed as a major risk as well as a major success factor in software projects [29], [32], [7].

This paper proposes a requirements engineering process based on the development approaches mentioned above: a process improvement model, the CMMI, *Capability and Maturity Model Integration* [30]; a process framework, the RUP, *Rational Unified Process* [20]; and a set of software development fundamental principles and practices, represented by the agile methods [5].

First, related work to the proposed process is summarized, followed by a brief description of the development approaches considered during the process conception: CMMI, RUP and agile methods and the rationale for their choice. Afterwards, the proposed requirements engineering process is briefly sketched, by describing its components, focusing on its activity flows. Orientations about the usage of agile methods practices are mentioned and the process compliance to CMMI is described. Finally, conclusions regarding this study are presented, commenting on agile principles application, CMMI compliance, contributions made by this paper and future work.

2. Related Work

In [8], a method is proposed for the definition of "*development strategies*" based on project risk analysis. The approaches indicated to form development strategies are: agile methods; plan-driven approaches (methodologies that emphasize planning); and methods based on the CMMI. The method proposed by Boehm and Turner is focused on strategies to solve risks. It is not dedicated to defining a specific development process, such as the requirements engineering process proposed in this paper.

The PMT, *Pattern-based Methodology Tailoring* approach, presented in [15], also uses risk analysis criteria to instantiate software development methodologies. In this approach, the instantiated development processes are formed by organizational patterns, recurrent solutions for human work organization observed in proved success projects. The specific features and risks associated to each project are used as inputs to determine of the organizational patterns that form the project development process. PMT does not define a specific process to be followed, focusing more on process patterns recommendations for a given project than on their implementation and integration.

Several other studies discuss the integration of the approaches studied in this paper – CMMI, RUP and agile methods – in different ways. Several authors study RUP compliance to CMMI, considering its principles and practices, and defining possible ways to complement it [33], [23], [14], [12], [31], [1]. Some studies analyze RUP and agile methods compliance, their common points and risk areas, as well as strategies to hybrid development processes [20], [21], [1], [2], [27], [9]. There are also studies that state that CMMI and agile methods may be used together, creating a synergy that makes it possible for an organization to benefit from both of them [18], [26], [25].

An approach described in [22] integrates different methodologies in order to define a development process. It is based on process frameworks as means to implement agile values and principles within an organization that has an institutionalized development process which is in compliance to RUP and CMMI. Focusing on values, principles and practices of agile methods, some features of RUP are explored to increase process productivity. Such approach is similar to the one used in this paper.

The approach described in [22] focuses on strategies for building the organization process framework whilst this paper focuses on the process itself, its activities, roles and artifacts, as well as its institutionalization within an organization, limited to requirements engineering.

3. Development Approaches

CMMI defines a set of goals and practices to be followed and executed during a development project. CMMI models are organized into process areas that group related goals to a specific context. An organization must progressively reach specific process area goals in order to reach compliance to CMMI. By reaching theses goals, the organization increases its maturity level, which varies from 1 to 5.

Two process areas are related to requirements engineering: *Requirements Management*, executed by maturity level 2 organizations, and *Requirements Development*, implemented by maturity level 3 organizations. The process defined in this paper intends to reach the goals of these two process areas.

RUP is a process framework in which the proposed process elements definition is based. It describes a series of activities, roles and artifacts that need to be selected according to each software project.

RUP process elements are organized into disciplines. The discipline directly related to this paper is the *Requirements* discipline, whose elements are partially included in the process proposed here.

After creating the proposed process using contributions from CMMI and RUP, an effort was made in order do include principles and practices from *agile methods* in the process. These methods use a *lightweight* software development process, without a strict definition of work products and activities, which values communication and interpersonal collaboration, the generation of tangible results and a the capacity to accommodate changes.

4. Requirements Engineering Process

The following sections describe the requirements engineering process proposed in this paper.

4.1. Overview

The requirements engineering process proposed here was conceived and institutionalized in a software development organization. It is part of a global development process created by this organization with the objective of significantly improve its development process productivity and the quality of its software products.

The global process follows the RUP lifecycle, composed by Inception, Elaboration, Construction and Transition phases. Each phase executes a set of activity flows, using RUP-based process components: roles, artifacts and activities.

The role concept is a description of behavior and responsibilities of a particular person or group. Behavior is described by activities associated to the role. Responsibilities are defined based on artifacts created, update and/or controlled by the role. An artifact is a portion of information that is produced, modified or used during a process. The activities describe orientations about what should be done by each role, producing the project artifacts.

The activity flows of the requirements engineering process are:

1. Define System Scope;
2. Refine Software Requirements;
3. Manage Changes.

Each one of these flows is formed by a set of activities. Each activity is executed by one or more members of the team that perform a certain role. Some activities are present in more than one flow, and are called *recurrent activities*.

4.2. Artifacts

The artifacts produced during the process execution are:

Input Documents: documents related to the project, possibly produced before the beginning of the process and often related to requirements.

Requirement Attributes: data gathered during the requirement analysis process: indicators and registering information, status and indexes such as importance for the business, relevance for the architecture, size (or complexity) estimates and development priority.

Traceability Matrixes: used to document dependency between requirements. It is possible do document traceability explicitly using tables, spreadsheets or requirements management tools, or implicitly, using other project artifacts.

Glossary: used to document common vocabulary used in the project, using client's terms. It is created in the beginning of the project and it evolves during the system development.

Vision Document: in this document it is defined the vision that all people related to the project have about the product that needs to be delivered, concerning main needs, features and acceptance criteria.

Software Requirements Specification (SRS): it captures a global vision of all requirements with a brief description of each one.

Requirement Functional Specification: it complements the system Software Requirements Specification (SRS), giving further information about a specific functionality. It describes with details the interaction between the actors (users or external systems) and the system that happens to fulfill the requirements.

Domain Model: model of the initial objects of the system or another representation of the essential entities of the system.

Interface Prototype: description of one or more interface with the system user; it may be, for example, a functional prototype, screen sample or even free-hand drawings.

Change Request: it documents the necessity of a change (defect, improvement or new requirements), information about impact, status and reasons for the decisions taken about implementing the change.

Project Repository: it has all the artifacts used in the software development process.

Approvals: official communications sent by the client, stating acceptance of a delivered artifact.

4.3. Recurrent Activities

Activities executed repeatedly during process work flows: *Manage Requirements* and *Assure a Common Vision*.

Manage Requirements activity contributes to scope management and change control of the project, and it involves the maintenance of the requirement attributes, including requirement development priority and traceability.

Requirement attributes are information such as size or complexity, business and system architecture importance, status and changes history. Traceability relationships represent dependencies between project requirements and artifacts, in such a way that a requirement change may imply on the need to change other related requirements or artifacts. Development priority is an attribute determined based on client's immediate needs, on system architecture relevance, on project risks, on requirement impact and on any other goal or restriction that is important for the project.

Assure a Common Vision activity addresses knowledge management between each person involved on the development process. This activity involves group requirement analysis, artifact revision by team members and by client, approval gathering and Glossary maintenance.

4.4. Define System Scope

It's goal is to define the problem to be solved by the system, identifying system needs, features, acceptance criteria and software requirements.

The activities to be performed are:

Understand Customer Requirements: customer requirements are stakeholder needs, expectations and constraints [29]. This activity involves: identifying project requirement providers; understanding the problem; defining system limits, identifying what is part of the system and what is not; identifying stakeholder needs and constraints and defining system features.

Understand Product Requirements: it means to refine customer requirements, defining system software requirements. These requirements may be functional or non-functional (usability, reliability, performance and support requirements among others). The suggested technique to identify and document these requirements is the one used in RUP: use case modeling [6].

Manage Requirements and *Assure a Common Vision* (described under the Recurrent Activities section of this paper).

4.5. Refine Software Requirements

The goal of this activity workflow is to refine requirements identified during scope definition. The workflow activities are:

Specify Software Requirement: to detail the software requirements found during the system scope definition, to refine analysis models and to update all other artifacts that need to be updates. Once more, the suggested technique is de use case modeling, specifying each use case using its description, event flows (basic and alternatives), business rules and user interface prototype.

Model Interface: to create a representation of the system interface with the user, including screen prototypes, storyboards, integration tests with interface tools and any other mechanism that give feedback about usability and performance of the system, and to validate the comprehension of its business rules.

Analyze the Domain: to create a domain model that represents the relation between the business objects of the system. It contributes with the activities of design and data modeling.

Manage Requirements and *Assure a Common Vision* (described under the Recurrent Activities section of this paper).

4.6. Manage Changes

Change management aims to register requirement changes, to analyze their impact and to decide about when to implement them. The activities considered here are (Figure 1):

Submit Change Request: it happens when a project participant, client or team member, notices the need for a change on system requirements. The reason for this change may be a problem on requirement analysis or business comprehension, incompatibility between requirements or even improvement opportunity. The perceived requirement change is described and submitted.

Complement Change Request: right after it has been submitted, a Change Request has only an initial description of desired changes. The next step is to complement the request by registering information that will be used to decide whether the changes shall be accepted or not. Complementing the Change Request means revising and further detailing the Change Request description and recording impact analysis considerations.

Analyze Change Request: after complementing the Change Request, one must decide whether the change will be implemented or not, considering project goals and risks and change impact and benefits. This activity represents the analysis done in order to take this decision. It includes registering the decision taken. If the Change Request is accepted, this activity triggers a project replanning (outside the scope of the activity).

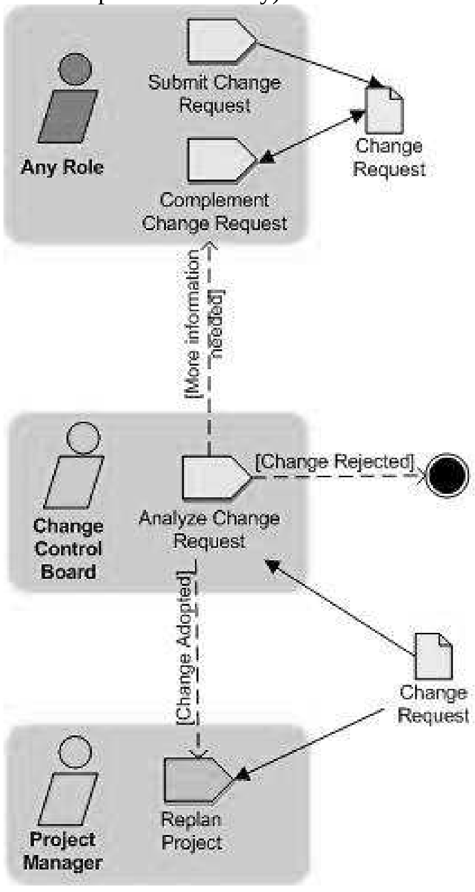

Fig. 1. Manage Changes

5. Considering Agile Practices

Even though there is a vast catalogue of development methods, techniques, and strategies proposed by the agile methods, it has been observed that these resources are difficult to insert into the requirements engineering process described in this paper. The reasons for this to happen may be the following:

1. Agile methods have simplified definitions and they have promoted project team self-organization. Therefore these methods do not define a set of elements specifically dedicated to requirements engineering. That makes it difficult to clearly identify how to apply agile methods within the context of a requirements engineering focused process.

2. Agile methods bring more advantages when their practices are used in synergy. Executing a software development process that employs only a portion of agile methods practices is less productive and more risky, once several techniques can only be well succeeded when applied together [13]. Since the process proposed on this paper is restricted to requirements engineering discipline, one must not assume that the techniques used in the other development process disciplines will follow agile methods. This context makes it difficult to insert agile method techniques in the proposed process.
3. This work has been developed within a specific organization whose typical projects do not include the characteristics indicated to the use of agile methods. Some of the characteristics of the projects developed in this organization are: large and heterogeneous teams; international customers; customer request for a certain level of formality in the project documentation.

Considering the items listed above, but still acknowledging the benefits that may be reached using agile methods, the following strategy was adopted: instead of adopting specific techniques, an approach based in values, principles and practices of the agile methods is used. These values, principles and practices may be used inside the activities of the proposed process.

The major points of agile practices in the development process activities are described below:

Face-to-face "conversation" and "communication" [19], [2], [4], "assume simplicity" [2], [4] and "model with others" [2] are practices that can be used in all process activities, notably in group requirement analysis held as part of the Assure a Common Vision activity of the proposed process.

"Self-organized teams" [28] may also be largely used, as long as each activity input and output are predefined and listed as part of process description. The team autonomy is used to determine techniques used to perform each activity.

"Active participation of stakeholders" [2] may be used during the process, and should involve "conversation" [19] and "use of simple tools" [2].

Agile Modeling practices may be used through the process, once a large number of those practices are strongly related to requirements engineering. In fact, it has been detailed described in [2] how to apply agile modeling practices and techniques within a project using unified process.

Finally, Manage Change activity promotes practices such as "embrace change" [4], [2] and "maximize stakeholder investment" [2], once stakeholders are part of the change control board that defines whether changes will be implemented or not.

6. CMMI Compliance

The table below describes how CMMI goals and practices associated to Requirements Management and Requirements Development process areas are reached by the proposed process activities.

Table 1. Fulfillment of CMMI Goals and Practices

REQM	**Requirements Management (ML2)**
SG1	***Manage Requirements***
SP.1.1	**Obtain an Understanding of Requirements**
Activity	Understand Customer Requirements Understand Product Requirements Assure a Common Vision
SP.1.2	**Obtain Commitment to Requirements**
Activity	Manage Requirements Manage Changes
SP.1.3	**Manage Requirements Changes**
Activity	Manage Changes
SP.1.4	**Maintain Bidirectional Traceability of Requirements.**
Activity	Manage Requirements
SP.1.5	**Identify Inconsistencies between Project Work and Requirements**
Activity	Assure a Common Vision
RD	**Requirements Development (ML3)**
SG1	**Develop Customer Requirements**
SP.1.1	**Elicit Needs**
Activity	Understand Customer Requirements
SP.1.2	**Develop Customer Requirements**
Activity	Understand Customer Requirements
SG2	**Develop Product Requirements**
SP.2.1	**Establish Product and Product-Component Requirements**
Activity	Understand Product Requirements Specify Software Requirement Analyze Domain
SP.2.2	**Allocate Product-Component Requirements**
Activity	Understand Product Requirements Specify Software Requirement Analyze Domain
SP.2.3	**Identify Interface Requirements**
Activity	Specify Software Requirement Model Interface
SG3	**Analyze and Validate Requirements**
SP.3.1	**Establish Operational Concepts and Scenarios**

Activity	Specify Software Requirement
SP.3.2	**Establish a Definition of Required Functionality**
Activity	Specify Software Requirement
SP.3.3	**Analyze Requirements**
Activity	Manage Requirements Assure a Common Vision
SP.3.4	**Analyze Requirements to Achieve Balance**
Activity	Understand Customer Requirements Assure a Common Vision
SP.3.5	**Validate Requirements with Comprehensive Methods**
Activity	Assure a Common Vision

7. Conclusions

7.1. Contributions

The main contribution represented by this study is describing a case study that brings together widely adopted software process development approaches that are usually employed separately: CMMI process goals and practices, RUP-based activity flows, roles and artifacts; and agile method best practices as part of process execution guidelines.

This case study might benefit organizations that are currently defining or improving their development process and it might be used to: comprehend alternatives to integrate the used approaches; acquire knowledge about how to work on development process points that might be difficult to define and institutionalize, particularly in a process based on CMMI, RUP and agile methods; understand how goals, practices and principles from distinct software development approaches might contribute to improving the organization adopted process.

The requirements engineering process proposed here also represents a contribution and might be used by organizations interested in improving its processes and results.

7.2. Future Work

Although this work has reached its final goal by institutionalizing the proposed development process, activities related to result evaluation and organization process improvement remain being executed. The process described in this paper continues to evolve in a controlled manner, adapting to characteristics of projects developed within the organization and constantly incrementing its guidelines and best practices repository.

Some future work opportunities are:

1. Complementing the requirements development process described here with a detailed requirement related metrics plan. After defining such a plan, those metrics must be collected in every project, creating historical data that will support future decisions related to the process.
2. Complementing the process by including a detailed analysis of CMMI generic goals and practices concerning maturity level 2 and 3. Such goals and practices were considered during this process elaboration but should be further explored and associated to each process activity.
3. Experimenting agile methods principles, practices and techniques in a variety of projects, using the proposed process, in order to identify specific process activities that might leverage such resources and integrate them into process description.

8. References

[1] AMBLER, S. W. Agile Modeling and the Unified Process. 2001. http://www.agilemodeling.com/essays/agileModelingRUP.htm

[2] AMBLER, S. W. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. New York: John Wiley & Sons, Inc., 2002.

[3] AVISON, D. E.; FITZGERALD, G. Where Now for Development Methodologies? *Communications of the ACM*, v.46. n.1, p. 79-82, January 2003.

[4] BECK, K. Extreme Programming Explained: Embrace Change. Addison-Wesley Longman, Inc.: 2000.

[5] BECK, K. et al. Manifesto for Agile Software Development. Agile Alliance. 2001. http://www.agilemanifesto.org/

[6] BITTNER, K.; SPENCE I. Use Case Modeling. Addison-Wesley Professional: 2002.

[7] BOEHM, B. Software Risk Management: Principles and Practices. IEEE Software, v.8, n.1, p.32-41, January 1991.

[8] BOEHM, B.; TURNER, R. Using Risk to Balance Agile and Plan-Driven Methods. *IEEE Computer*. v.36, n.6, p. 57-66, June 2003.

[9] BOOCH, Grady.; Martin, Robert C; Newkirk, James. The Process. Preliminary chapter of Object Oriented Analysis and Design with Applications. 2d. ed.: Addision Wesley Longman: 1998. http://www.objectmentor.com/publications/RUPvsXP.pdf

[10] CLEGG, C. et al. The Performance of Information Technology and the Role of Human and Organizational Factors. tech. report. Economic and Social Reseach Council: UK: 1996.

[11] FIRESMITH, D; HENDSRSON-SELLER, B. The OPEN Process Framework: An Introduction. Addison-Wesley Professional, December 2001.

[12] FITZGERALD, B.; RUSSO, N. L.; O'KANE, T. Software Development Method Tailoring at Motorola. *Communications of the ACM*, v.46, n.4, p. 65-70, April 2003.

[13] FOWLER, M. Is Design Dead?. May, 2004. http://www.martinfowler.com/articles/designDead.html

[14] GALLAGHER, B.; BROWNSWORD, L. The Rational Unified Process and the Capability Maturity Model – Integrated

Systems/Software Engineering. RUP/CMMI Tutorial : ESEPG: Carnegie Mellon University: 2001.

[15] HARTMANN, J.; PRICE, R. T. Utilizando padrões organizacionais e avaliação de risco para adaptar a metodologia de desenvolvimento de software. Dissertação de Mestrado: Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Graduação em Computação: Porto Alegre: 2004.

[16] ISO 9000-3. ISO 900-3 Guidelines for the Application of ISO9001 to the Development, Supply, and Maintenance of Software. Int'l Standards Organization, Geneva, 1991.

[17] ISO/IEC 12207. ISO/IEC 12207 Information Technology – Software Life-Cycle Processes. Int'l Standards Organization, Geneva, 1995.

[18] JEFFRIES, R. Extreme Programming and the Capability Maturity Model. XProgramming.com, XP Magazine: 2000. http://www.xprogramming.com/xpmag/xp_and_cmm.htm

[19] JEFFRIES, R. Essential XP: Card, Conversation, Confirmation. 2001.
http://www.xprogramming.com/xpmag/expCardConversationConfirmation.htm.

[20] KRUCHTEN, P. The Rational Unified Process: An Introduction. 2nd ed. Addison-Wesley: March de 2001.

[21] KRUCHTEN, P. Agility with the RUP. Cutter IT Journal, The Journal of Information Technology Management. Cutter Consortium. v.14, n.12, p. 27 – 33, December 2001.

[22] LYCETT, M.; MACREDIE, R. D.; PATEL, C.; PAUL, R. J. Migrating Agile Methods to Standardized Development Practice. *IEEE Computer*. v.36, n.6, p. 79-85, June 2003.

[23] MANZONI, Lisandra; PRICE, Tom. Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3. In *IEEE Transactions on Software Engineering*: vol. 29: no. 2: 2003: pp.181-192.

[24] PAULK, Mark C. et. al. Key Practices of the Capability Maturity Model, Version 1.1. Technical Report: CMU/SEI-93-TR-025: 1993.

[25] PAULK, Mark C. Extreme Programming from a CMM Perspective. *IEEE Software*: November/December 2001: pp. 19-26.
http://www.agilealliance.org/articles/articles/XPFromACMMPerspective-Paulk.pdf

[26] REIFER, D. J. XP and the CMM. IEEE computer.org – *IEEE Computer*: 2003.
http://www.computer.org/software/homepage/2003/s3man.htm

[27] SANTOS, J. B. Extraindo o melhor de XP, Agile Modeling e RUP para melhor produzir software. Vice-Reitoria Administrativa, PUC-Rio: November 2002.

[28]. SCHWABER, K. The Impact of Agile Processes on Requirements Engineering. 2002.
http://www.agilealliance.org/articles/schwaberkentheimpacto/view?searchterm=requirements

[29] SEI. Capability Maturity Model Integration (CMMI), Version 1.1, CMMI for Software Engineering (CMMI-SW, V1.1), Staged Representation. August 2002. http://www.sei.cmu.edu/cmmi/models/sw-staged.doc

[30] Software Engineering Institute. Process Maturity Profile, CMMI v1.1, SCAMPI v1.1 Class A Appraisal Results, 2005 Mid-Year Update. CMMI Appraisal Program. Carnegie Mellon University, Pittsburgh, PA, USA: September 2005.

[31] SMITH, J. Reaching CMM Levels 2 and 3 with the Rational Unified Process. White Paper: Cupertino: CA: 2000.

[32] SOMMERVILLE, I. Requirements Engineering, An Overview. Class Presentation. Lancaster University, UK: 2001.

[33] TYSON, B; BROWNSWORD, L.; BROWNSWORD, R. Leveraging RUP and CMMI for Real-World Successes. Software Engineering Institute – Carnegie Mellon University, Number Six Software: 2004.
http://www.numbersix.com/csdi/documents/ESEPG-CMMIandRUP-Final.pdf