Mapping RUP Roles to Small Software Development Teams

Pedro Borges[1], Paula Monteiro[2], and Ricardo J. Machado[2]

[1] CIICESI, Escola Superior de Tecnologia e Gestão de Felgueiras do Instituto Politécnico do Porto, Portugal
pmb@estgf.ipp.pt
[2] Dept. Sistemas de Informação, Universidade do Minho, Guimarães, Portugal
{pmonteiro,rmac}@dsi.uminho.pt

Abstract. In the last decades the complexity of software development projects had a significant increase. This complexity emerges from the higher degree of sophistication in the contexts they aim to serve and from the evolution of the functionalities implemented by the applications However, many software corporations have a reduced dimension (micro, small or medium) which imposes a considerable constraint to the number of individuals that might be involved in each project. This limitation has obvious consequences to the individual's efficiency and effectiveness. In this paper we describe a Rational Unified Process (RUP) tailoring to simplify the number of RUP roles. With this tailoring we obtain one set of RUP roles that, without neglecting any critical role of the software development process, may easily be adopted by a small or medium software development team. In this paper, we present and justify a complete set of mapping rules between RUP roles and one possible configuration for small software development teams.

Keywords: RUP, small software teams, SME, RUP tailoring.

1 Introduction

Now more than ever, the software development industry is being put to the test, as a joint result of several stress factors. First, we have been witnessing a significant increase in the complexity inherent to software development projects, due not only to a higher degree of sophistication in the contexts they aim to serve, but also to the natural evolution of the out-of-the-box features offered by the myriad of available technologies and software systems. On the other hand, the ever growing importance of reducing time-to-market decreases the error margins, boosting the pressure applied on the teams to deliver better software in less time.

In order to react to this scenery and tip the playing field on their behalf, eastern corporations have responded by establishing partnerships with software factories based on developing countries and, in some cases, by creating their own off-shore software development centers. However, though these might be good solutions for large scale corporations and projects, they are inappropriate for some SMEs (Small

S. Biffl, D. Winkler, and J. Bergsmann (Eds.): SWQD 2012, LNBIP 94, pp. 59–70, 2012.

and Medium Enterprises) [1], given the usually short-term nature of their projects and the considerably time-consuming specification requirements.

Since, SMEs urge for methodologies with the potential to help them cope with the challenges faced, arouse from the low level of process standardization, RUP (Rational Unified Process) [2] presents itself as a useful reference, given the wide set of roles proposed to structure software development teams. However, there's a lack of RUP configurations suited for micro (employing less than 10 people) and small companies (employing less than 50 people) [1]. This paper aims to help small scale organizations by providing them a RUP configuration that, without neglecting any critical function of the software development process, may easily be adopted during a project's execution period. In order to do so, the roles proposed by RUP will be thoroughly reviewed in order to select a much smaller subset of key participants that will inherit the duties of the suppressed roles [3].

The following sections are organized as: section 2 provides an overview of RUP tailoring approaches. Section 3 presents a RUP tailoring to SMEs, Section 4 presents a role mapping to the model presented and Section 5 presents a brief discussion about the roles accumulation. Section 6 presents the conclusions and future work.

2 Related Work

The Rational Unified Process (RUP) [2] is a well known software process development framework which extends the Unified Process [4] which in turn resulted from the integration and evolution of older processes such as Rational Approach [5] and Objectory Process [6].

RUP is presented as a disciplined approach for assigning tasks and responsibilities within an organization, with the aim of ensuring the production of high quality software that meets the needs of their users and in strict compliance with a predictable timetable and budget. This framework defines a set of activities, roles and artifacts, which need to be selected according to the software project. Each project is performed by a group of employees having one or more roles assigned. Each role participates in one or more activities and, as result, of their engagement in each activity one or more artifact is produced. Currently, this framework comprises more than eighty artifacts, one hundred and fifty activities and about forty roles.

Although RUP is widely used its structure lacks flexibility, and small enterprises that adopt it have to face a very long development cycle, and an "overload" of documentation when using it mechanically [7, 8]. To overcome the excess of documentation and the high cost of a long development cycle while, at same time, maintaining quality (or at least not reducing it too much), the software process must be tailored (by adding, merging and/or deleting activities, roles and artifacts).

Another set of research efforts [9-11] arose from the conclusions of a study presented in [12]. They consider that leaving the responsibility of tailoring RUP to each project context will cost too much time and resources; leading them to give to the teams, before the project start, an adapted version of RUP.

The work presented in [13] conveys a very pragmatic view about how RUP can be configured to "speed up" its adoption (of course without missing any procedural component considered essential) and thus prove the possibility of its successful

adoption in SME contexts. In this way, the path followed was to perform a significant simplification of the list of artifacts to produce, followed a cost/benefit analysis of each of the artifacts provided by the methodology.

Following a completely different approach, [14] presents one RUP configuration primarily oriented to organizations that develop software in a process-oriented way, which may be appropriate for small. The authors present a set of business modeling artifacts whose production is considered essential. In [15] the authors continue this guideline, by analyzing further the business modeling artifacts, and presenting a way to set up a methodology that can incorporate procedural improvements to, thereby, enable organizations that adopt RUP to get a better ranking on the CMM scale.

According to [16], RUP is much too complex and sophisticated to be capable of being implemented as a successful practice. It is alleged that RUP does not frame in the best way the existing roles and that does not adequately involve the users during the transition phase.

The authors of [17, 18] present extensions to RUP in order to make it compliant with CMM, in particular with maturity levels 2 and 3. These works start to analyze the gaps between RUP and CMM and then they propose some activities and artifacts that will complement RUP.

Agile Methods (AM) are attempting to offer once again an answer to the impatient business community asking for lighter weight and at same time faster software development processes [19]. Some agile practices are used to change the team roles, like for instance, cross-functional and self organizing teams [19, 20].

Some studies discuss the integration of RUP and Agile Methods [21, 22]. They explain how RUP and Agile Methods can be used in conjunction. According to the author it is relatively easy to the RUP users to adopt AM practices by tailoring RUP.

RUP, CMMI and Agile Methods can also be used together [23]. The authors describe the components of requirements engineering process, and present the process compliance with CMMI. They also give some orientations to the usage of agile practices in the requirements engineering process.

There are several other research efforts that propose the simplification of RUP, by adopting tailoring techniques. However, none of them consider the organizational context existent in SMEs, namely from the roles point of view. This paper addresses this perspective.

3 Tailoring RUP for Constructing the Base Model

Next, we will describe the Base Model which corresponds to a RUP role simplification tailored to medium-sized companies, which essentially seek a software process that helps to design and implement solutions with high levels of quality (perceived by the customer) and to deal with the complexity inherent in projects of medium/high scale.

To achieve the proposed goals, we conducted a detailed analysis of the RUP roles in order to identify the roles considered to be essential, by satisfying at least one of three conditions. These conditions are defined in [3].

System Analyst (C1, C2): Scope management is vital to the success of any project, otherwise different opinions between customers and suppliers are almost inevitable

during the project execution. Therefore, the participation of system analysts is paramount since the beginning, to identify and document with the utmost rigor the requirements (functional or nonfunctional), in order to allow the supplier to correctly estimate the effort involved in performing the project.

Is extremely important that the person who performs this role has appropriate training, focusing mainly on two aspects: first, beyond basic knowledge in management, it is desirable to understand in detail the client's business domain and to perceive the real motivations and relevance of the requirements presented by the client; second, in order to develop his activities as appropriate and in accordance with the best practices, it is desirable that the person has been trained and presents some practice of requirements engineering techniques and methods (Software Requirements in SWEBOK [24]).

User-Interface Designer (C2): The scope of this role intervention in a project varies according to the nature of the artifacts to be developed. Despite not being able to consider this role as a critical one, it is a fact that its best performance largely depends on a strong background in specific areas of knowledge (such as the software ergonomics). This domain cannot be considered widespread by the software engineering professionals.

Database Designer (C2): Like the previous role, database designer is also considered essential to the process, mainly due to the specificity of its knowledge. Although database design techniques are mandatory in any computer science degree, usually the depth of the acquired skills is insufficient. It is desirable that the performer of this role has (at least) the following competencies: configuration and optimization of database engines; advanced knowledge on setting up indexes, views, and constraints; advanced knowledge on the implementation of triggers and stored procedures; and advanced knowledge on standardization of data models.

Implementer (C1): Regardless of how good the architecture and design of a software solution is, it will not be successful without the involvement of the implementer (usually called programmers) of the sub-systems and components that support the desired functionalities.

Integrator (C1, C2): In SMEs, it is usual to find several Implementers involved in project, each one engaged with a specific set of tasks. To ensure that the work runs in a smoothly and efficiently way, it is essential to have someone responsible for maintaining the implementers aware of the project context, for identifying the tasks to be undertaken and for appointing the person responsible for each one. This person must also decide how the individual tasks will be integrated and incorporated in the final outcome of the project. An example is the definition of the interfaces between the various sub-systems. Besides these technical tasks, the integrator is also responsible for the initial definition of the critical dates of the project and for developing a plan for the integration of the sub-systems, to allow the project manager to inform the client when each feature is expected to be available.

This role requires the capability to monitor the activities of each implementer to ensure to adopt measures to minimize the impact in case of failure. Appropriate training should include knowledge of human resource management, allowing the encouragement and empowerment of their work. It is desirable that the person has been trained and presents some practice in the SWEBOK knowledge areas of software engineering management and software engineering process.

Software Architect (C1, C2): The performer of this role is responsible for setting the technologic foundation on which the project implementation should be based. When this context is imposed by the client, the software architect is responsible for estimating the technical risks involved and how they might be mitigated. Other responsibilities are: the definition of the skeleton of the system to be created; the characterization of the components (defining the functions and boundaries of each one); the advising on the frameworks that should be adopted. This role requires basic training in software architecture and design (SWEBOK knowledge area of software design) and also the capability of monitoring of the market trends to be aware of the most appropriate tools and frameworks for achieving the goals of a given project.

Process Engineer (C1, C2, C3): It is considered essential the existence of a person mainly concerned with the management of the development process, its adaptation to the organizational context and monitoring its implementation, in order to identify and implement possible process improvements. This role requires a detailed knowledge of the adopted development process (in this case RUP).

Project Manager (C1, C2): This is an important role because it has the responsibility to assume a global overview of the project through a detailed interaction with the internal and external participants. The project manager must create the conditions for the project to achieve success, by ensuring the timeliness and the fulfillment of all the undertaken commitments. To perform this role with quality it is important to have training in several areas such as: basic knowledge in management; knowledge about the client's business domain; project management methodologies (like PMBOK); negotiation skills.

Project Reviewer (C3): This role cannot be considered critical to the project success and it does not require any specific skills besides the ones required for any of the other roles. However, it is important that the responsible for this role presents a good knowledge about the business domain. Due to certain responsibilities related to the verification and approval of several artifacts produced by other participants, and concerned about conflict of interests, the person that performs this role should not accumulate with another role within the project.

Test Manager (C1, C3): It is essential the existence of a role whose major responsibility is to ensure the product quality by devising a plan for internal quality audits and coordinating its implementation. This is another role that should not be cumulated with other roles in particular with those roles related to the design and construction phase.

System Tester (C1, C3): The implementation of the audit plan is performed by the system testers which may be entrusted with very different tasks, like the artifacts documental review and testing behavior. This is an essential role to the process.

Course Developer (C2): The main concern of this role is the preparation and coordination of training. The person with this role needs competencies in the area of didactics and pedagogy as main skills to execute it.

System Administrator (C1, C2): In software development projects it is extremely important to have a person focused on ensuring the satisfaction of the infrastructure needs inherent to the process, particularly regarding: the personal computers of each element of the project team, according to the specific needs of each one; servers that

support the activities of the team; servers that support the testing procedures and quality assessment; servers that support the service provision to the outside. This person must have training in the following skills: practice in the SWEBOK knowledge area of software configuration; system administration; configuration and optimization of engine databases; negotiation and contracting of IT services.

4 Mapping RUP Roles into Base Model Roles

Rather than the 39 original roles proposed by RUP, as we can see by the stated in the previous section, it is feasible to reduce to 13 the number of the essential roles to

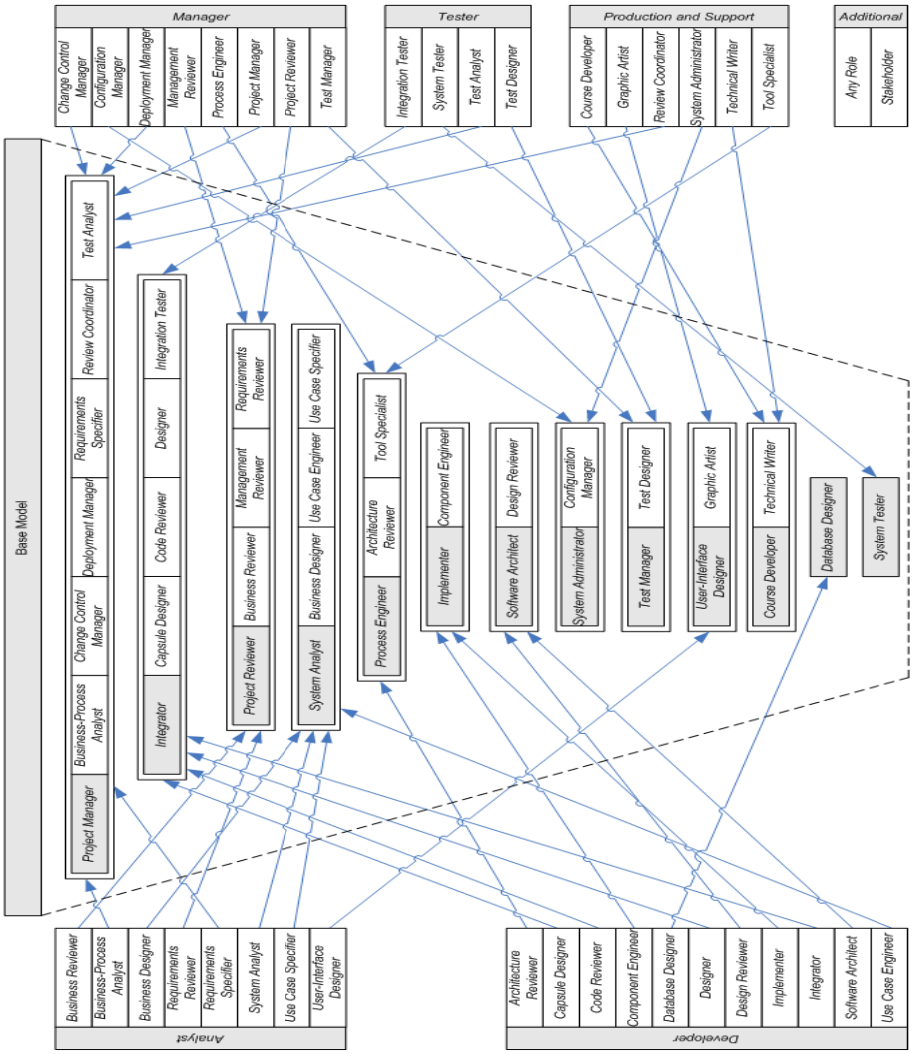

Fig. 1. Mappings of Base Model

implement a software process (that we call the Base Model) in the context of small development teams. However, the fact that any of the remaining 26 roles have not been regarded as essential to the process does not mean that we may discard their responsibilities or that they are not considered important for the effective and efficient implementation of the process. Instead, we propose a mapping of the remaining roles into each role previously considered essential, according to a set of guidelines defined in [3]. In figure 1, we present the Base Model roles and the mappings between RUP roles and roles considered essential in the Base Model. Next, we will present the proposed mappings and the respective justification.

Business Reviewer, Requirements Reviewer and Management Reviewer maps into Project Reviewer: The *project reviewer* is characterized by RUP as someone responsible for evaluating the project artifacts on specific key moments with power and legitimacy to, if necessary, suspend its execution. Therefore, this role can only be performed by someone with a high level of responsibility and authority within the organization, possibly even at the highest level, since it is not unusual that managers of SMEs be personally involved in supervising and monitoring projects. Thus, *project reviewers* have the appropriate profile and therefore are able to evaluate the business artifacts (commercial proposals, business models, etc.) produced during the project (usually performed by the *business reviewer*).

The *requirements reviewer* has the responsibility to formally review the requirements identified and incorporate them into the use case model by the *requirements specifier*. Therefore, it is essential that this person is knowledgeable of the business domain and it should also be familiar with the modeling techniques used in the description of requirements. This role can be merged with the *project reviewer*.

The *management reviewer* and the *project reviewer* are separated by a slight difference, since both are defined as responsible for the review and evaluation of the artifacts produced at certain key moments. Thus, and assuming what was previously mentioned that the *project reviewer* must be someone with some knowledge in the business domain, the separation of these roles is not justified.

Business-Process Analyst, Requirements Specifier, Change Control Manager, Deployment Manager, Test Analyst and Review Coordinator maps into Project Manager: Since the *project manager* should be the person with greater proximity to the customer, he may assume the responsibility to define the business architecture and to describe their needs (in the form of business use cases), replacing the *business-process analyst*. This would ensure that the *project manager* knows in detail the scope of the project, which allows him to deal appropriately with the requests for changes and carry out an effective monitoring of its implementation.

The *project manager* is also responsible for ensuring compliance with the scope and minimizing the contract extensions that do not provide value to the organization. However, the effectiveness of his intervention may suffer if he does not have a thorough understanding of the settled goals between the parties. For these reasons, becoming responsible for the activities of the *requirements specifier* increases his control over the project, by taking responsibility for listing and characterizing the requirements, properly managing the customer expectations and ensuring they are implemented by the project team.

In small-scale projects, it is normal that the *change control manager* responsibilities are assumed by the *project manager* or the *software architect*. However, despite being essential that the *software architect* possesses a deep and updated knowledge about the project, it is considered more important that the *project manager* ensures an effective control over it; otherwise unrealistic expectations would be created upon the client. The *project manager* can also assume such responsibilities.

The *deployment manager* plans and coordinates the transition to the user's community of the products resulting from the development efforts. The success of this type of activity largely depends on the dialogue with the client and on the planning and communication skills of the person involved. The person performing this role must work closely with the *project manager*, enough reason to merge these two roles.

The reason to map the *test analyst* into the *project manager* arises from the fact that the testing efforts must to be aligned with the project context and the needs of end users. The *project manager* is the person with more knowledge about the scope of the project and about how the artifacts will be used. In dialogue with the *integrator* and *test designer*, the *project manager* defines the scenarios that must be tested. In addition, he can also be responsible for: monitoring the progress of the testing process; analyzing the results produced by the *test designer* and ensuring that they are reported to the respective *integrators* and that they are timely corrected; evaluating the effectiveness of the testing process through the perceived quality reported by the end users; facilitating the communication between *test designer's* and *integrator's*.

Besides all the previous roles, the *project manager* is also considered to be the best role to coordinate and control the activities inherent to role *review coordinator*.

Business Designer, Use Case Specifier, Use Case Engineer maps into System Analyst: The *business designer* is responsible for detailing the specification of the business solution, producing artifacts that characterize the business entities involved, their expectations and interactions. However, despite being focused on business, the activity of this role is closer to the *system analyst* responsible for the identification and documentation of the project requirements. Since, the *system analyst* must also know the business domain, we recommend merging it with the *business designer*.

The *use case specifier* role interacts closely with the end users and works together with the *system analyst* in the description of the use cases that embody the identified requirements. Since this role is not defined as having their own specific tasks but only acts as an assistant, he should be merged with the *system analyst*.

The *use case engineer* is responsible for ensuring that one or more embodiments of the use cases represent, in a coherent and comprehensive way, the project's requirements. Therefore, and given the tasks performed by the *system analyst* during requirements elicitation, we considered that the merge with the *use case engineer* activities is a natural extension of his work.

Architecture Reviewer and Tool Specialist maps into Process: The *process engineer* role supports the project methodology and is responsible for monitoring its implementation and making the necessary adjustments to optimize its effectiveness.

The *architecture reviewer* is explicitly a technical role, since he formally reviews the architecture designed by the *software architect*, in order to validate the design choices. It is important that the *architecture reviewer* has the necessary legitimacy to point out mistakes or omissions. Apart from the obvious technical skills, it is

important to have good communication skills, enabling to manage any conflicts with the required sensitivity and delicacy. The *process engineer* is the person in best suitable for accumulating the *architecture reviewer* responsibilities. By the nature of his function, he has the necessary legitimacy to evaluate the performance of everyone involved in the software development, on which he should have extensive experience.

In what concerns the *tool specialist* (which includes the identification of stakeholders needs regarding the tools to assist/facilitate their work and the selecting the most appropriate applications to meet their needs) we have decided to map his responsibilities into the *process engineer* role.

Capsule Designer, Code Reviewer. Designer and Integration Tester maps into Integrator: The *capsule designer* has a profile similar to the *designer* but more focused on the accomplishment of the components performance requirements. Thus, given the similarity of both profiles, the *integrator* role assumes those responsibilities.

When coordinating the software development efforts of a team, the *integrator* needs a good level of technical expertise, which means that he should also be confident in assuming the responsibilities of the *code reviewer*, by reviewing and auditing the code source produced by the *implementers*. This means, the *integrator* must also verify if each component has been implemented in accordance with his instructions and detect potential problems.

The *designer* must translate the architecture conceived by the *software architect* into a coherent solution of components/modules/sub-systems to be implemented, detailing responsibilities, operations and relations between them. Taking into account that the *integrator* is responsible to ensure a successful integration of several existing components, it makes sense that both roles are performed by the same person, maximizing the efforts involved in the design stage.

Considering that the primary responsibility of the *integration tester* is to perform the integration tests, which are essential to verifying that the various components that make up the solution are working well together, this role can be considered as a natural extension of the activities conducted by the *integrator.*

Component Engineer maps into Implementer: The *component engineer* is focused on the design of the internal structure of each sub-system, particularly regarding with operations, methods, attributes, relationships and requirements of each design class. This role is strictly related with the *implementer's* duties.

Design Reviewer maps into Software Architect: The RUP methodology strongly suggests the existence of roles especially devoted to review artifacts written by third parties. Therefore, in order to maintain the independence, we suggest that the role of the *design reviewer* is accumulated by the *software architect*.

Configuration Manager maps into System Administrator: We suggest merging the responsibilities of the *configuration manager* and the *system administrator*. If the *system administrator* is already responsible for managing and providing the infrastructure used by the project team, it makes sense to also perform the configuration management.

Test Designer maps into Test Manager: From the activities carried out by the *test analyst* emanates a set of test scenarios that constitute a starting point for the work to be performed by the *test designer*, which is responsible for coordinating the planning,

design and implementation of the necessary tests. However, these activities are a consequence of the responsibilities that the literature attributes to the *test manager*, described as the main role responsible for the success of the testing effort. Therefore, the *test manager* must assume the test designer role.

Graphic Artist maps into User-Interface Designer: The performance of the *graphic artist* benefits from a refined aesthetic sensibility and some experience in the use of image manipulation applications. However, since it is common that these characteristics are also presented in the *user-interface designer*, this role can also be the responsible for meeting the project needs of image and graphic communications.

Technical Writer maps into Course Developer: Given the restrictions usually found in smaller organizations, the *implementer* role could assume the responsibilities of the *technical writer*. However, the RUP methodology suggests that the production of content support (user manuals, etc.) could be performed by different persons from those involved in the technical execution of the project. Since the activities of the *course developer* can be regarded as complementary to those of the *technical writer* (the training material he produces is also directed to end users) we recommended the merging of these two roles.

Table 1. Roles Accumulation Restrictions (Inside Project)

Should not accumulate with... / If it is...	Project Manager	Integrator	Project Reviewer	System Analyst	Process Engineer	Implementer	Software Architect	System Administrator	Test Manager	User-Interface Designer	Course Developer	Database Designer	System Tester
Project Manager		x	x										
Integrator	x								x				?
Project Reviewer	x												
System Analyst													
Process Engineer									x				
Implementer									x				?
Software Architect									x				?
System Administrator													
Test Manager		x			x	x	x			x		x	
User-Interface Designer									x				?
Course Developer													
Database Designer									x				?
System Tester		?				?	?			?		?	

5 Accumulation of Roles

After we have justified, in general, the Base Model role set it is relevant to analyze the eventual restrictions to the accumulation of several roles by the same team member. This recommendation may result in the mobilization of a higher number of resources to perform a project. However, it is also a fact that it could contribute to a

better performance of each role, while avoiding ethical incompatibilities. In table 1, we present the identified restrictions of roles accumulation inside the same project.

The symbol "x" indicates an absolute restriction, which means that this accumulation must be avoided. For instance, the integrator should not accumulate with the test manager role because there are some ethical issues involved, since the test manager evaluates the artifacts produced under the integrator's supervision. The symbol "?" indicates a conditional restriction; i.e., an accumulation of roles that might be possible if some regards are considered. As an example, the integrator presents one conditional restriction to accumulate the system tester role, because this accumulation will only be possible in the cases where the system tester tasks are not performed within the same software development line under coordination as integrator. The detailed explanation of these restrictions will be presented in a future work.

6 Conclusions

The Rational Unified Process is a comprehensive software development process, which aims to help organizations to efficiently use resources at their disposal to ensure the effective implementation of the goals they want to achieve. However, the lack of an appropriate RUP configuration for SMEs (small and medium sized companies) that develop software has justified our effort to propose a reduced set of roles involved in the implementation of the RUP methodology. As a result, we have suggested the Base Model, which is a tailoring approach of RUP composed by 13 roles. The other 26 RUP roles not considered in the Base Model have been mapped into the Base Model roles according a set of presented guidelines. In this study we do not discuss how one person can handle all the activities of each performed role, in a reasonable time. However this is a relevant issue and it will be analyzed in a future work.

As future work, we will develop a Reduced Model that will simplify further the set of RUP roles to be adopted in small settings. This simplification will be performed by considering part of the Base Model roles as non essential and the other part as essential. The roles considered as non essential are mapped into the essential roles. We will assess the Reduced Model with a case study of a CMMI level 3 certified team [25, 26].

References

1. European Commission. Small and Medium-sized Enterprises Definition, vol. 2011 (2005)
2. Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley (2003)
3. Borges, P., Monteiro, P., Machado, R.J.: Tailoring RUP to Small Software Development Teams. In: 37th Euromicro Conference, SEAA 2011 (2011)
4. Jacobson, I., Booch, G., Rumbaugh, J.: The unified software development process. Addison-Wesley (1999)
5. Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J., Houston, K.: Object-oriented analysis and design with applications, 3rd edn. Addison-Wesley (2007)
6. Jacobsen, I.: Object Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley (1992)

7. de Barros Paes, C.E., Hirata, C.M.: RUP Extension for the Development of Secure Systems. In: ITNG 2007, pp. 643–652 (2007)
8. Jieshan, L., Mingzhi, M.: A Case Study on Tailoring Software Process for Characteristics Based on RUP. In: CiSE, pp. 1–5 (2009)
9. Hanssen, G.K., Westerheim, H., Bjørnson, F.O.: Tailoring RUP to a Defined Project Type: A Case Study. In: Bomarius, F., Komi-Sirviö, S. (eds.) PROFES 2005. LNCS, vol. 3547, pp. 314–327. Springer, Heidelberg (2005)
10. Westerheim, H., Hanssen, G.K.: The introduction and use of a tailored unified process - a case study. In: 31st EUROMICRO Conference, SEAA 2005, pp. 196–203 (2005)
11. Hanssen, G.K., Bjørnson, F.O., Westerheim, H.: Tailoring and Introduction of the Rational Unified Process. In: Abrahamsson, P., Baddoo, N., Margaria, T., Messnarz, R. (eds.) EuroSPI 2007. LNCS, vol. 4764, pp. 7–18. Springer, Heidelberg (2007)
12. Hanssen, G.K., Westerheim, H., Bjørnson, F.O.: Using Rational Unified Process in an SME – A Case Study. In: Richardson, I., Abrahamsson, P., Messnarz, R. (eds.) EuroSPI 2005. LNCS, vol. 3792, pp. 142–150. Springer, Heidelberg (2005)
13. Hirsch, M.: Making RUP agile. OOPSLA 2002 Practitioners Reports (2002)
14. Fernandes, J.M., Duarte, F.J.: A reference framework for process-oriented software development organizations. Software and Systems Modeling 4, 94–105 (2005)
15. Duarte, F.J., Fernandes, J.M., Machado, R.J.: Business Modeling in Process-Oriented Organizations for RUP-based Software Development. In: Reference Modeling for Business Systems Analysis, pp. 98–117. Idea Group Publishing (2006)
16. Hesse, W.: Dinosaur meets Archaeopteryx? or: Is there an alternative for Rational's Unified Process? Software and Systems Modeling 2, 240–247 (2003)
17. Manzoni, L.V., Price, R.T.: Identifying extensions required by RUP to comply with CMM levels 2 and 3. IEEE Transactions on Software Engineering 29, 181–192 (2003)
18. Chang, G.: Modifying RUP to comply with CMM levels 2 and 3. In: ICISE 2010, pp. 1–5 (2010)
19. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile Software Development Methods: Review and Analysis. Technical Research Centre of Finland, Espoo, Finland (2002)
20. Ana Sofia, C.M., Felipe, S.F.S., Arnaldo, D.B.: Mapping CMMI Project Management Process Areas to SCRUM Practices. In: Proceedings of the 31st IEEE Software Engineering Workshop, pp. 13–22. IEEE Computer Society (2007)
21. Ambler, S.W.: Agile Modeling and the Rational Unified Process (RUP), vol. 2011 (2001)
22. Ambler, S.: Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. John Wiley & Sons, Inc., New York (2002)
23. Cintra, C.C., Price, R.T.: 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. In: Alencar, F.M.R., Sanchez, J., Werneck, V. (eds.) Workshop em Engenharia de Requisitos, Rio de Janeiro, pp. 153–159 (2006)
24. Abran, A., Bourque, P., Dupuis, R., Moore, J., Tripp, L.: Guide to the Software Engineering Body of Knowledge - SWEBOK. IEEE Press (2004)
25. Monteiro, P., Machado, R.J., Kazman, R.: Inception of Software Validation and Verification Practices within CMMI Level 2. In: Fourth International Conference on Software Engineering Advances, ICSEA 2009, pp. 536–541 (2009)
26. Monteiro, P., Machado, R.J., Kazman, R., Henriques, C.: Dependency analysis between CMMI process areas. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 263–275. Springer, Heidelberg (2010)