Tailoring RUP to Small Software Development Teams

Pedro Borges
CIICESI, Escola Superior de Tecnologia e Gestão de Felgueiras do Instituto Politécnico do Porto
Porto, Portugal
pmb@estgf.ipp.pt

Paula Monteiro and Ricardo J. Machado
Dept. Sistemas de Informação
Universidade do Minho
Guimarães, Portugal
pmonteiro@dsi.uminho.pt, rmac@dsi.uminho.pt

***Abstract*— In the last decades 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 features implemented by the available software systems and applications. However, the reduced dimension of many software corporations imposes a significant constraint to the group of individuals that might be involved in each project, with obvious consequences to their efficiency and effectiveness. This paper describes how to accomplish a configuration of the Rational Unified Process (RUP) in order to 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 during the project execution period.**

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

I. 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. Finally, the rise of strong international players based on developing countries (like India, China, Pakistan, etc) boosted competitiveness, given their technological maturity (attaining the highest levels of the CMM – Capability Maturity Model [1] scale) and cost advantages (due to the considerably lower wages).

Since, SMEs (Small and Medium Enterprises) [2] 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) [3] presents itself as a useful reference, given the wide set of roles proposed to structure software development teams. However, in spite of its alleged easy customization process, there's a lack of RUP configurations suited for micro (employing less than 10 people) and small companies (employing less than 50 people) [2]. Thus, 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 were thoroughly reviewed in order to select a much smaller subset of key participants that will inherit the duties of the suppressed roles.

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

II. RELATED WORK

The Rational Unified Process (RUP) [3] 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. 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]. To overcome the excess of documentation and the high cost of a long development cycle while, at same time, maintaining (or at least not reducing it too much) quality, the software process must be tailored. This means that the software process must be modified by adding, merging and/or deleting activities, roles, artifacts and other elements.

The need of tailoring a software process based on RUP to decide what process elements best suit the company or project gave origin to a metamodel for process tailoring compliant with RUP. This metamodel extends the RUP model by adding a set of elements and relationships, and a set of well-formed rules used to guide the process tailoring activities [8].

The work presented in [9] conveys a very pragmatic view about how RUP can be configured to "speed up" its adoption (of course without missing any procedural component

978-0-7695-4488-5/11 $26.00 © 2011 IEEE
DOI 10.1109/SEAA.2011.55

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, [10] presents one RUP configuration primarily oriented to organizations that develop software in a process-oriented way, which may be appropriate for small entities that do not justify the existence of a functional structure. This paper also highlights a possible need for internal restructuration in organizations that adopt the RUP in order to overcome their difficulties in the composition of the set of roles involved.

According to [11], 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. In [12, 13] another alternative approach is quantitatively compared with RUP regarding the underlying concepts of both approaches as evidenced in their meta-models. Also according to this article, RUP is considered negligent on the most appropriate way to manage the human resources involved in their use.

Other studies discuss the integration of RUP and Agile Methods [14, 15]. They explain how RUP and Agile Methods can be used in conjunction. Some studies show how Agile Methods can help organizations to accomplish CMM and CMMI goals [16, 17]. RUP, CMMI and Agile Methods can also be used together: in [18], a requirements engineering process based on CMMI, on RUP and on a set of agile principles and practices is presented.

III. TAILORING RUP FOR CONSTRUCTING THE BASE MODEL

Any company, regardless of its size, has an organic structure (more or less formal) that identifies the roles of each employee, defines its areas of intervention and establishes the responsibilities they have to assume, in order to achieve the organization's objectives. Thus, it is common to define organizational charts and assigning specific positions to its employees.

RUP features nearly forty roles relevant to the software development process, assigning to each one a particular set of specific responsibilities. However, after a brief analysis of the applicability of this set of roles in the context of an SME in the software development sector we can conclude that the overwhelming majority of the cases they do not have a number of employees as high as the number of participants expected. Therefore, even considering that in the context of a given project, a person can be appointed to several roles, an excessive accumulation of responsibilities may become too complex or demanding.

To be applicable in a simpler context, a software development process must be based in the involvement of a lower number of players with different responsibilities. However, a linear process to adapt the set of roles defined by RUP to the specific players existing in a particular organization may find some difficulties resulting from the characteristics of each role. In order to be more efficiently, this process should take place in three separate stages:

1. Reduce the number of relevant roles in the process of software development, in order to make it easier to understand and therefore to apply. Each of the roles proposed by RUP should be examined in order to assess whether they are essential (and should be kept in its essence) or not (and its tasks/responsibilities should be assigned to the remaining essential roles);
2. Identify some restrictions to the accumulation of tasks/responsibilities by the same person taking on the roles obtained in the previous step;
3. Propose one mapping between the previously identified essential RUP roles and some of the canonical roles that one can usually find in an SME of the software development sector. This mapping should not be considered a dogma; it should be reviewed in the context of each SME that wishes to adopt it, according to its specific characteristics.

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.

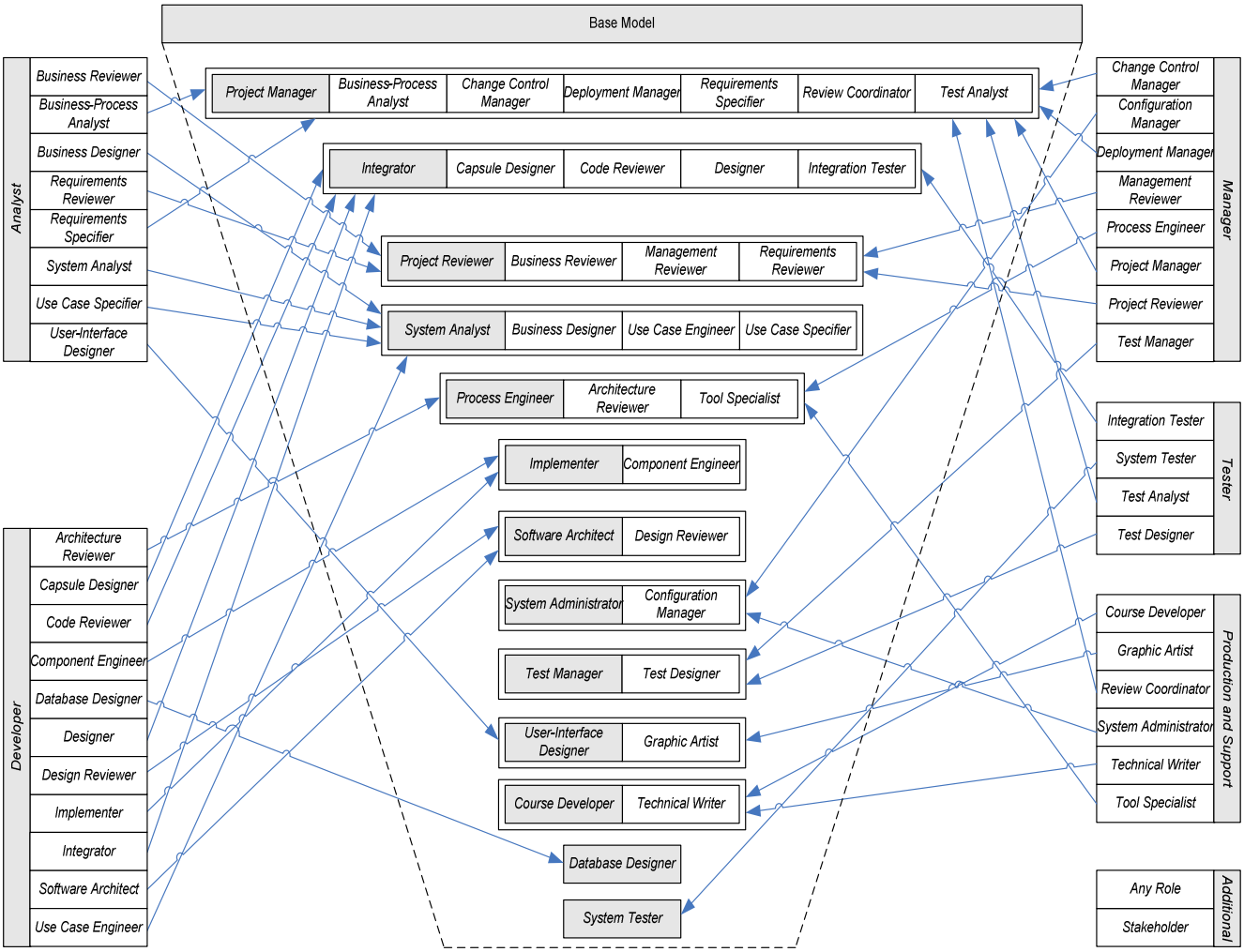
However, it should be recalled that the reduction in the number of people involved in the process of software development inevitably results in a higher criticality of individual performances, since it is necessary that each one assumes a much larger range of responsibilities, usually without possessing more time or resources to perform them. This increases not only the stress of each team member, as well as the probability of committing an error (or omission) than if there were a greater number of people involved in the process. Furthermore, it is not always easy to find the necessary skills in the people who are available to perform the activities previously assigned to the suppressed roles.

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 the following conditions:

- C1: If the role is suppressed the project will definitively fail;
- C2: The role demands a set of unique skills completely different from those skills demanded by other roles;
- C3: The role imposes potential conflicts of interest when merged with another role.

Taking into account the previous conditions, we suggest that the following roles are considered essential and thus integrate our Base Model:

- System Analyst (C1, C2);
- User-Interface Designer (C2);
- Database Designer (C2);
- Implementer (C1);
- Integrator (C1, C2);
- Software Architect (C1, C2);
- Process Engineer (C1, C2, C3);



Mappings of Base Model

- Project Manager (C1, C2);
- Project Reviewer (C3);
- Test Manager (C1, C3);
- System Tester (C1, C3);
- Course Developer (C2);
- System Administrator (C1, C2).

As an example we will present the justification of why a given role should be considered essential. Looking into the *Process Engineer* we can see that this role is essential because it satisfies the three conditions C1, C2, C3. It is considered essential the existence of a person mainly concerned with the management of the development process, with the process adaptation to the organizational context and with the monitoring implementation of the process, in order to identify and implement possible process improvements. This role requires a detailed knowledge of the adopted development process (in this case RUP). Finally, it is important to promote the independence of this role regarding the other roles, to ensure the legitimacy required to make adjustments and corrections when needed.

IV. 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 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 one of roles previously considered essential, according to the following guidelines:

- the appropriate profiles for performing both roles should be easily compatible;
- the responsibilities of both roles should, whenever possible, find themselves framed in the same (sub-) area of expertise;
- if the responsibilities of both roles are framed in distinct (sub-) areas of knowledge, their accumulation

by the same person will result in positive synergies; if the elected recipient of a specific mapping is not in the best position to accumulate it (either because it is already responsible for too many roles or because the involved roles require a strong commitment), another mapping should be tried; although not being the first choice, it will gather better conditions to ensure its effective execution.

In figure 1, we present the Base Model roles and the mappings between RUP roles and roles considered essential in the Base Model. Considering figure 1, we can identify the following mappings:

- Business-Process Analyst, Requirements Specifier, Change Control Manager, Deployment Manager, Test Analyst and Review Coordinator maps into Project Manager;
- Capsule Designer, Code Reviewer. Designer and Integration Tester maps into Integrator;
- Business Reviewer, Requirements Reviewer and Management Reviewer maps into Project Reviewer;
- Business Designer, Use Case Specifier, Use Case Engineer maps into System Analyst;
- Architecture Reviewer and Tool Specialist maps into Process Engineer;
- Component Engineer maps into Implementer;
- Design Reviewer maps into Software Architect;
- Configuration Manager maps into System Administrator;
- Test Designer maps into Test Manager;
- Graphic Artist maps into User-Interface Designer;
- Technical Writer maps into Course Developer.

As an example, we can justify the why the *Architecture Reviewer* and the *Tool Specialist* are mapped into the *Process Engineer*:

(1) The *process engineer* is a role that supports the project methodology and is responsible for monitoring its implementation and making the necessary adjustments to optimize its effectiveness.

(2) 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 better conditions to accumulate the *architecture reviewer* responsibilities, because, 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.

(3) 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.

V. 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.

As future work, we will develop a Reduced Model that will simplify further the RUP role set. This Reduced Model will be suitable for micro companies.

REFERENCES

[1] M. C. Paulk, C. V. Weber, B. Curtis, and M. B. Chrissis, The capability maturity model: guidelines for improving the software process: Addison-Wesley Longman Publishing Co., Inc., 1995.

[2] E. Commission. (2005, 2011-02-24). Small and Medium-sized Enterprises Definition. http://ec.europa.eu/enterprise/policies/sme/facts-figures-analysis/sme-definition/index_en.htm

[3] P. Kruchten, The Rational Unified Process: An Introduction, 3 ed. Boston: Addison-Wesley, 2003.

[4] I. Jacobson, G. Booch, and J. Rumbaugh, "The unified software development process," ed: Addison-Wesley, 1999.

[5] G. Booch, et al., Object-oriented analysis and design with applications, third edition: Addison-Wesley Professional 2007.

[6] I. Jacobsen, Object Oriented Software Engineering: A Use Case Driven Approach: Addison-Wesley Professional, 1992.

[7] L. Jieshan and M. Mingzhi, "A Case Study on Tailoring Software Process for Characteristics Based on RUP," in CiSE 2009, pp. 1-5.

[8] E. B. Pereira, R. M. Bastos, and T. C. Oliveira, "A Systematic Approach to Process Tailoring," in ICSEM '07, 2007, pp. 71-78.

[9] M. Hirsch, "Making RUP agile," presented at the OOPSLA 2002 Practitioners Reports, Seattle, Washington, 2002.

[10] J. M. Fernandes and F. J. Duarte, "A reference framework for process-oriented software development organizations," Software and Systems Modeling, vol. 4, pp. 94-105, 2005.

[11] W. Hesse, "Dinosaur meets Archaeopteryx? or: Is there an alternative for Rational's Unified Process?," Software and Systems Modeling, vol. 2, pp. 240-247, 2003.

[12] B. Henderson-Sellers, G. Collins, R. Dué, and I. Graham, "A qualitative comparison of two processes for object-oriented software development," Information and Software Technology, vol. 43, pp. 705-724, 2001.

[13] B. Henderson-Sellers, R. Due, I. Graham, and G. Collins, "Third generation OO processes: a critique of RUP and OPEN from a project management perspective," in APSEC 2000, 2000, pp. 428-435.

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

[15] P. Kruchten, "Agility with the RUP," Cutter IT Journal, vol. 14, pp. 27-33, 2001.

[16] D. J. Reifer, "XP and the CMM," Software, IEEE, vol. 20, pp. 14-15, 2003.

[17] M. C. Paulk, "Extreme programming from a CMM perspective," Software, IEEE, vol. 18, pp. 19-26, 2001.

[18] C. C. Cintra and R. T. Price, "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," presented at the Workshop em Engenharia de Requisitos, Rio de Janeiro, 2006.