A Reduced Set of RUP Roles to Small Software Development Teams

Paula Monteiro
Centro ALGORITMI, Escola de Engenharia
Universidade do Minho
Guimarães, Portugal
pmonteiro@dsi.uminho.pt

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

Ricardo J. Machado and Pedro Ribeiro
Centro ALGORITMI, Escola de Engenharia
Universidade do Minho
Guimarães, Portugal
{rmac, pmgar}@dsi.uminho.pt

***Abstract*—Software projects are always increasing their complexity. The complexity of projects arises due to the increased sophistication of software applications and of their implemented features. However, most of the projects are developed by small organizations. Since these companies have a reduced dimension, the number of individuals that constitute each software development teams will also be significantly reduced. This paper describes a Rational Unified Process (RUP) configuration composed by a reduced set of RUP roles. This configuration may easily be adopted by a small software development team during the project execution period. Additionally, we have characterized each role in this reduced model by identifying the corresponding activities in charge and by creating individual sheets detailing their responsibilities. An initial assessment of the effectiveness of this RUP configuration was performed using CMMI-DEV maturity level 2 (ML2) as a reference model.**

***Keywords*-RUP; small teams; SME; RUP tailoring**

I. INTRODUCTION

In the last decades we have been witnessing a significant increase in the complexity inherent to software development projects. This complexity is due to the natural evolution of the features offered by the innumerous available technologies and software systems and due to the higher degree of sophistication of the project. The constantly growing need of reducing time-to-market decreases the error margins, increasing the pressure applied on the teams to deliver better software in less time. The rise of strong international organizations from developing countries (like India, China, Pakistan, etc) increased competitiveness, due to their technological maturity (most of them achieved the highest levels of the CMM – Capability Maturity Model [1] scale) and due to the cost advantages (because of the considerably lower wages).

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. These might be a good solution for large scale corporations and projects. However, they are inappropriate for some SMEs (Small and Medium Enterprises) [2], given the usually short-term nature of their projects and the considerably time-consuming specification requirements.

As a consequence of the above issues, SMEs urge for methodologies with the potential to help them cope with the challenges faced. Those challenges arouse from the low level of process standardization that SMEs usually uses. RUP (Rational Unified Process) [3] presents itself as a useful reference, given the wide set of roles proposed to structure software development teams. Despite its alleged easy customization process, RUP lacks of configurations suited for micro (employing less than 10 people) and SME (employing less than 50 people [2]). So, with this paper we aim to help these 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, we will simplify the configuration of RUP roles described in [4, 5].

In [4, 5], we described the *base model*, which is a tailoring approach of RUP composed by 13 roles (essential roles). We conducted a detailed analysis of the RUP roles in order to identify the roles considered to be essential. The remaining 26 "non-essential" roles 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 proposed a mapping of the remaining roles responsibilities into one "essential" role, according to a set of defined guidelines. All essential roles and mappings between the non-essential roles and the essential are presented in [5]. In Fig. 1, we can see the *base model* roles and the non-essential roles mapped to each essential role.

In this paper, the *base model* roles will be thoroughly reviewed in order to select a much smaller subset of key participants in a small software development team, by considering some of the previous essential roles as non-essential. The remaining essential roles will inherit the duties of the suppressed roles. This new model will be evaluated by analyzing the CMMI ML2 accomplishment when we use our RUP configuration.

The following sections are organized as: Section 2 provides an overview of RUP tailoring approaches. Section 3 describes and justifies our approach to get a RUP configuration to SMEs (that we call *reduced model*); Section 4 presents a complete description of the responsibilities of

two roles; in Section 5 we briefly describe the case study we have developed to initially assess the effectiveness of this RUP configuration using CMMI-DEV ML2 as a reference model. Section 6 presents the conclusions and future work.

II. Related Work

In the last years several software process development frameworks have been presented and implemented. One of the most well-known frameworks is the Rational Unified Process (RUP) [3]. This is an iterative software development process which assigns tasks and responsibilities within an organization, to ensure the production of high quality software (meeting the needs of their users in strict compliance with a predictable timetable and budget). RUP framework defines three basic elements: activities, roles and artifacts. A set of activities, roles and artifacts need to be selected according to the software project. Each project is performed by a group of actors having one or more roles assigned. Each role participates in one or more activities and as result of each activity one or more artifact is produced. This software development process is composed by more than eighty artifacts, one hundred and fifty activities and about forty roles.

Despite RUP being 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 [6, 7]. Tailoring the software process was a way to overcome the “overload” of documentation and the high cost of a long of development cycle while the quality is maintained or slightly reduced. This means that the software process must be modified by adding, merging and/or deleting activities, roles, artifacts or other elements.

The conclusions of a study presented in [8] gave origin to set of research efforts described in [9-11]. In these studies it is considered that leaving the responsibility of tailoring RUP to each project context will cost too much time and resources. To overcome this issue the teams should use an already adapted version of RUP before they start each software project.

A metamodel [12] for process tailoring compliant with RUP arose from the need of tailoring a software process based on RUP to decide what process elements best suit the company or project. 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.

The work presented in [13] presents how RUP can be configured to "speed up" its adoption, without missing any procedural component considered essential, and thus prove the possibility of its successful adoption in SME contexts. The author starts to perform a significant simplification of the list of artifacts to produce, followed by a cost/benefit analysis of each of the artifacts provided by the methodology.

In [14], RUP is considered highly 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 [15, 16] 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 work, RUP is considered negligent on the most appropriate way to manage the human resources involved in their use.

Some studies [17-21] present extensions to RUP in order to make it compliant with CMM and CMMI, in particular with ML2 and ML3. To extend RUP, these works analyze the gaps between RUP and CMM or CMMI and then propose activities and artifacts that will complement RUP to allow the compliance with the other models.

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 [22]. There are several examples of AM: Crystal [23], Agile Modeling [24], Scrum [25] and Extreme Programming [26]. Some of the agile practices are used to change the team roles, like for instance, cross-functional and self-organizing teams [22, 27]. In the cross-functional teams approach project team is divided into several small groups with the necessary know-how to perform a set of roles. In the self-organizing team approach, followed for instance by Scrum, the team is organized by itself instead of being organized by the *project manager*.

The integration of RUP and AM are studied in [24, 28-30]. In those works it is explained how RUP and AM can be used in conjunction. According to the author it is relatively easy to the RUP users to adopt AM practices. Since RUP could be tailored by the users to meet their needs, the merge between RUP and AM practices is easy to make.

RUP, CMMI and AM can also be used together [31]. In this study the authors present a requirements engineering process based on CMMI, on RUP and on a set of agile principles and practices. They describe the components of this requirements engineering process and the process compliance with CMMI. Regarding the AM, the authors give some orientations on the usage of agile practices in their requirements engineering process.

Several of the research efforts discussed in this section propose the simplification or extension of RUP, by adopting tailoring techniques. However, none of them consider the organizational context existent in SMEs, namely from the roles point of view. In this paper we address this perspective.

III. Reduced Model

Despite the effort already carried out to get a mapped sub-set of the original RUP roles [4, 5], the resulting *base model* is still difficult to be directly adopted by SMEs. Therefore, we have considered to be appropriate the seeking for a model involving a smaller number of roles, giving up of some specializations and promoting versatility. However, to achieve this smaller set of roles, it is not adequate to remove some roles and randomly distribute their responsibilities by the remaining ones. In doing so, the balance achieved with the *base model* would be deprecated in the resulting *reduced model*, implying the failure of its execution due to the

inability of one or more individual in perform their responsibilities. Instead, it is proposed to carry on the simplification process according to the following guidelines: (1) identifying which roles previously considered "essential" may have lesser prominence when compared with the others; (2) identifying the role in better conditions to assume the responsibilities of each excluded participant, considering his profile; (3) validating if the proposed mappings will not unduly increase the intervention area of the destination roles, to ensure that they have real conditions to responsibly assume the responsibilities of the various tasks to be performed.

In the *base model* some of the recommended mappings were between suppressed roles into one role that in the *reduced model* described in this paper will be eliminated as an autonomous role. Therefore, it is crucial to define new mappings. In addition, it is required not only to validate if the new mappings do not overload too much any remaining role, but also to ensure that they do not jeopardize the independence that should exist between some role holders. In cases in which this occurs, a new mapping should be proposed in order to balance the responsibilities and workload of each essential role.

However, it is necessary to be conscious that the possible easiness of applying the *reduced model* when compared with the *base model* is usually achieved with a quality reduction in the artifacts produced and/or a higher production cost (due to the less experience/specialization of the people involved). Nevertheless, these disadvantages can be regarded as a minor evil in organizational contexts, in which the only alternative would be the use of a much more ad-hoc process without roles and responsibilities formalization which often results in a greater waste of resources and inconsistency of actions.

In Fig. 1, we present a comparative table between the *base model* (from [4, 5]) and the *reduced model* (proposed in this paper). In this table it is visible the level of simplification performed in the *reduced model* when compared with the *base model*. By analyzing the table, we can see the elimination (as autonomous roles) of the following essential roles: *system analyst*, *software architect*, *user-interface designer*, *course developer* and *database designer*. The responsibilities of those roles were mapped into the remaining roles. Next, we will present the proposed mappings and the respective justification.

A. System Analyst, Business Designer and Use Case Specifier Maps into Project Manager

According to RUP, one of the *system analyst* responsibilities is to coordinate the requirements elicitation process in order to delimitate the project scope.

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

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

Figure 1. Comparative table between the *base model* and the *reduced model*

However, his intervention should be monitored and coordinated by the *project manager*, since he is the person closest to the client and that needs to be constantly informed about the work in progress. Furthermore, in SMEs is common to assign the *project manager* role to a person with Software Engineering background (or even with Requirements Engineering background). Therefore, we believe that we can eliminate the *system analyst* and at same time impose a greater involvement of *project manager* in the requirements elicitation process. In some cases, however, the *system analyst* can optionally be maintained in coexistence with the project manager.

The *business designer* intervention intends to improve the work of the *business-process analyst,* in order to characterize properly and thoroughly a part of the client organization. Therefore it can be considered a supporting role to the *business-process analyst* activities. So, we consider acceptable to map this role into the *project manager*, extending in a natural way its intervention since he is also responsible for the *business-process analyst* tasks.

The *use case specifier* role interacts closely with the end users and work 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 follow the *system analyst* and be merged with the *project manager*.

B. Software Architect, Database Designer, Course Developer and Designer Reviewer Maps into Integrator

Although cyclical, the involvement of the *software architect* role in the development process is meaningful in its beginning, namely in the draft and detail phases. Therefore, in smaller organizations, it is difficult for these professionals to claim their value since it is hard to make their work profitable in the subsequent project phases. Usually the project size and/or complexity of such organizations do not justify the need for the *software architect* role. Frequently it is not possible to provide to these professionals the resources required (time for research, training, infrastructure, etc.) to follow efficiently the emergence of relevant technological developments. So, maintaining this role will be artificial, resulting inevitably in its neglect during the methodology operationalization. Thereby, we propose the mapping of *software architect* into the *integrator* whenever necessary, because he shares the permanent need for technological updating, and because the person chosen to assume this role will be the most technically experienced inside the organization. However, there are disadvantages associated to this simplification: possible loss of coherence in software architecture activities, conducted either in separate projects or by different *integrators* within the same project, due to the absence of an external intervener to the project team that will act as a reference and help each *integrator* to find the best solution to the technological issues; possible decrease in the capacity of organizational learning and consequently of the innovative potential of the organization. These pros and cons, justify that the *software architect* role can optionally be maintained in coexistence with the project manager.

The majority of the undergraduate degrees address technical training (more or less advanced) on modeling and design of database models. Although some more advanced concepts (like triggers, stored procedures, etc.) are only addressed in the context of a specific database engine/ *technology*. Nevertheless, academic training usually does not cover more advanced topics (such as administration, backup strategies and data recovery, and optimization of the database engine/ *technology*). Thus, organizations seeking to have this particular knowledge need to apply for specialized training and usually associated to a specific database engine. However, the majority of professionals in this area of knowledge possess the minimum know-now required to perform this task. Therefore, we consider that, under the ongoing simplification effort, the *database designer* could be mapped into the *integrator* or the *implementer*. However, the fact that the *integrator* has a broader view of the project compared to the *implementer* can provide him the necessary capability to design a data model that includes not only all the current needs but also the improvements probably requested in a near future. Thus, we propose the mapping of the *database designer* into the *integrator*.

The quality of the support material to the user training is extremely important for the adoption of a new software application. In SMEs it is not necessary to maintain the *course developer* as an essential role, since, although probably not have training in the educational and training area, the *integrator* shall have all the conditions to produce support material to clarify the end-users about the use and operation of the new software application. The support material produced shall be evaluated and approved by another person in order to identify and correct possible gaps before its delivery to the end-users.

In the *base model* the *design reviewer* was mapped into the *software architect*. However, in the *reduced model* the *software architect* can optionally be considered a "non-essential" role. Thereby, it is necessary to identify another role capable of assuming the *design reviewer* responsibilities. The most reasonable solution is to map this role into the *integrator*, likewise to what was proposed to the *software architect*.

C. User-Interface Designer, Designer, Graphics Artist and Technical Writer Maps into Implementer

Although undoubtedly very important to the success of a project, the attractiveness and usability of the implemented user interfaces are issues that present lower priority when compared with others (like time management and risk management). This is why the *user-interface designer* is a natural candidate to be mapped into other role. Therefore, it is proposed that each *implementer* will also assume this role, because they usually are involved in the implementation of user-interfaces, even if those interfaces were not designed by them. So, it is normal that within his responsibilities and from the interaction with the *user-interface designers*, the *implementers* will learn the most important principles to

observe, and consequently replace the *user-interface designers*.

In the *base model*, the *designer* role was mapped into the *integrator*; assigning the *designer reviewer* role to the *integrator* results in a mismatch of responsibilities. Obviously, this situation is highly undesirable, since it will concentrate on the same person the system design responsibility and the evaluation of its correctness, destroying the process independence. On the other hand, the *reduced model* maps several additional roles to the *integrator*, which means that there is a tendency to the degradation of operational effectiveness, as a result of its huge range of responsibilities. For the above reasons, in order to minimize these problems, we propose to map the *designer* into the *implementer*, which will enable him to actively participate in the system design.

The performance of the *graphic artist* benefits from an accurate 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. However, since the *user-interface designer* was mapped into the *implementer*, the *graphic artist* should also be mapped into this role.

By suppressing the *course developer* as an essential role, the *technical writer* can no longer delegate to this role the production of support material and user manuals, with the aim to release the *implementers* for development activities. However, in small size teams, it is essential to enhance the knowledge of all participants. Therefore, since the contents to be produced by the *technical writer* emanates mainly from the implementation details controlled by the *implementers*, it is recommended to map the *technical writer* into this role.

D. Use Case Engineer Maps into Test Manager

In the *base model,* the *use case engineer* was mapped into the *system analyst* role. As we have already discussed, in the *reduced model* the *system analyst* role can optionally be mapped into the *project manager* role. Following the same rationale used to propose the *system analyst* mapping, the *use case engineer* should be mapped into the *project manager* role. However, we propose the mapping of the *use case engineer* role into the *test manager*, basing our decision on the following reasons:

- the *test manager* gathers all the technical conditions required for the proper performance of this role;
- as a result of the simplification process implicit in the *reduced model*, the *project manager* will be responsible for ten different roles, combined with the fact that he is the major responsible for the project, resulting in a huge load of responsibilities. So, it is more realistic to consider the *test manager* in better conditions to perform this role successfully;
- this mapping will increase the exhaustive knowledge of the requirements by the *test manager* which will allow an easily test plan preparation, and help the *project manager* to monitor the implementation.

IV. CHARACTERIZATION OF SOME ESSENTIAL ROLES

In the previous section, we carried out the exercise of reducing the number of RUP roles existent in the base model to make plausible the process application in the context of an SME. For this, we propose the mapping of roles established for the base model into a more limited set of eight distinct roles.

However, the operationalization of this synthesized set of roles lacks a detailed definition of their responsibilities in order to delimit the area of intervention of each participant. The characterization of the essential roles allows an easier selection of the person with the most suitable profile for each role and also describes to each individual what is expected from his intervention. However, since some of the organizations that are interested in adopting this set of roles may not be familiarized with RUP, we will present them independently of RUP terminology.

Accordingly, in order to contextualize the interactions that take place between the individuals, Fig. 2 identifies the most common communication flows. The diagram reveals that, within the same project, several lines of development may evolve concurrently.

However, it is not possible to implement any project without the existence of a significant interaction between the provider entity (in this case, a small software development company) and the client entity, consolidating multiple communications flows. Nonetheless, the existence of several contingent communications among the several internal and external stakeholders, the main link between the inside and outside of the organization should be the *project manager*, since only in this way can be ensured the effective control over the project execution.

Next, we present two examples of role descriptions detailing their responsibilities. Since we are presenting the *reduced model*, the role descriptions will describe the responsibilities of each role taking into account this model. These individual sheets do not intend to define the tasks of each role but to help their owners in their daily work, making it easier to remember what they have to do in each moment. So, it is natural that some small duties do not appear in these sheets, mainly if it is not translated into a concrete task that has to be performed at a given time. Despite presenting only two examples of role descriptions, we have all the individual sheets to the *reduced model* roles. These sheets were all used in the case study. In this paper, we have decided to include the descriptions of the roles *integrator* and *implementer* due to the fact that they form a pair of roles that work together and because they present a set of diverse responsibilities interesting to be presented here.

A. Integrator

This is certainly one of the most important roles of the proposed models. He is responsible to coordinate the production activities of the artifacts needed to achieve the objectives established by the *project manager* for a given system.

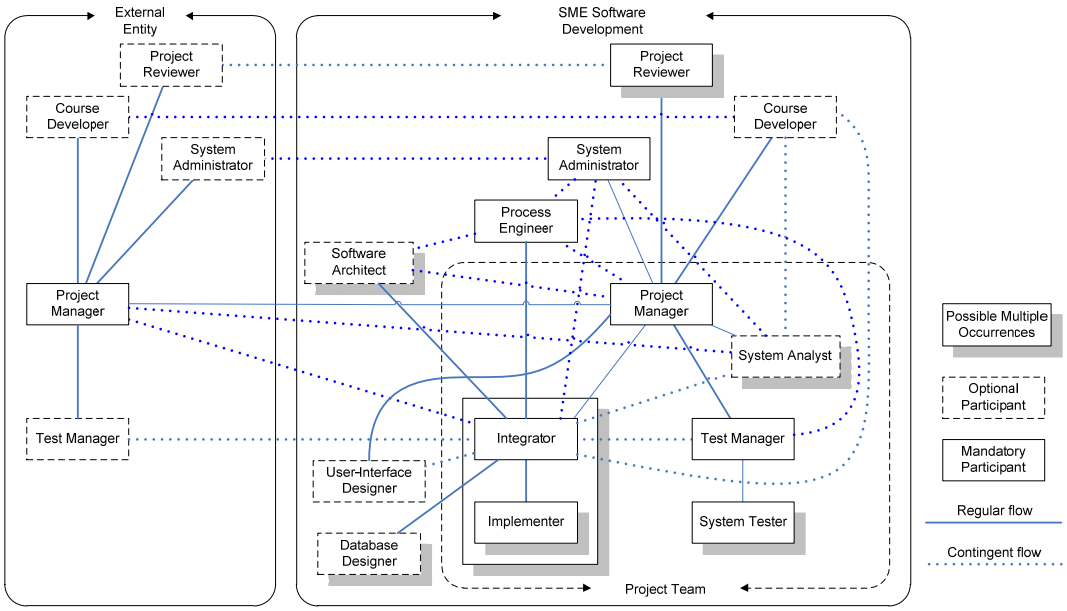

Figure 2. Communication Flow (Internal and External)

Thus, he must not only to assign tasks to the *implementers*, but also to monitor periodically their implementation in order to detect (as early as possible) any delays or problems, which should be solved in cooperation with the *project manager*. Additionally, he should cooperate with the *test manager* in order to provide all the necessary assistance to the evaluation of the artifacts available within his developments scope. His intervention is crucial to address two other aspects: the *integrator* should be able to overcome, by virtue of his experience, the eventual inexistence of technical knowledge from the *project manager* to efficiently guide the activities of the *implementers* allocated to the project; the high number of *implementers* involved in the implementation of large projects would make virtually impossible for one person (specifically the *project manager*) to coordinate and control their work. It is possible to overcome this problem by distributing the *implementers* by several lines of development, each one coordinated by a different *integrator*. The activity of this role is primarily focused within the organization, although it may need to interact directly with the outside world, mainly in projects where the external entities have a quality control team. The *integrators* play an important role in the operationalization of the reusing strategy of the organization source code, since they are responsible to ensure the reuse of the maximum number of existent components and to promote the creation of new components for general purposes with potential relevance for future projects.

1) Responsible for: **R1:** Ensuring compliance with all the defined objectives within his developments scope.
R2: Coordinating, as efficiently and effectively as possible, the *implementers* work, assigning them tasks best suited to their profile and avoiding situations of idle dependencies. In this sense, he should plan (as early as possible) the activities to develop throughout each iteration, defining timely and unequivocally the responsibility for implementing each task.
R3: Proposing to the *project manager* the number and duration of the iterations to perform, their content and their artifacts, along with the technical characteristics of each.
R4: Planning and execute the integration of the components implemented within his developments scope, in order to produce the required version in each iteration.
R5: Planning and performing the appropriated integration tests for each produced version, to ensure that the components included in the same version work properly together.
R6: Proposing to the *process engineer* the content, format and location of the internal documentation to be produced during the project.
R7: Ensuring timely production and publication of internal documentation considered necessary.
R8: Classifying each request of change received as *simple* (can be embedded in the current iteration without prejudice the other features) or *complex* (involves reduction of quality/functionality or increase of time or cost).
R9: Alerting (as early as possible) the *project manager* to situations where it is not possible to achieve all the iteration objectives within the time scope.
R10: Proposing to the *process engineer* the list of components to reuse and non-existent that can be created within his developments scope.
R11: Keeping, in collaboration with the *project manager*, a record of most meaningful events related to the project development (for instance, external entities delays, artifacts acceptance, etc.).
R12: Managing the communication between the *project manager* and the *implementers* that he coordinates.

R13: Evaluating, jointly with the *process engineer*, the performance of the *implementers* that he coordinates.
R14: Identifying and reporting to the *project manager* the project infrastructure needs (for the several environments to be established).
R15: Executing certification/production requests and ensuring the creation of supporting documentation.
R16: Ensuring the project availability in the development environment.
R17: Getting aware about the implementation details within his developments scope and be capable to discuss with other stakeholders (internal or external).
R18: Periodically inspect the source code generated by the *implementers* that he coordinates and assess their quality and compliance with the defined policies for the project.
R19: Standardizing, in accordance with the *process engineer*, the working methods of his team, in particular regarding to: tools (planning, coordination, document management, development, modeling, logging, unit testing and bug tracking) to be used; development methodologies and standards; package/components names; version control system location; code review process; integration build usage; error codes; log file formats and categories to be used; configuration settings (file, database, etc.) of each component within his developments scope.
R20: Ensuring that the *implementers* that he coordinates adopt the best practices of software development, namely: do not use hard-coded values, opting instead by its inclusion in the component/application configuration; implementation of unit testing in the components developed by them; frequently update (periods less than one week), in the control version system, the source code of the components in which he is involved with.
R21: Following the execution of the internal (and possibly external) audit quality plan on the artifacts produced within his developments scope.
R22: Periodically check the holidays calendar of the *implementers* that he coordinates.
R23: Ensuring the production, delivery and preservation of the required documentation to support the certification/production execution process related to his development scope.
R24: Notifying the *project manager* whenever he intends to make a holidays change.
R25: Evaluating the *implementers*' proposal for the interfaces of the main components of the system.
R26: Proposing to the *process engineer* the major technical decisions, namely: list of tools, application servers and database engines to be used; UML deployment diagram that describes how the system is interconnected with all the other relevant systems; UML component diagram describing the main components available in his development scope and identifying the relationships between them.
R27: Identifying, estimating and reporting to the *project manager* and *process engineer* the project technical risks resulting from the adopted architectural decisions.
R28: Preparing the indexes, views, constraints, triggers and stored procedures required to optimize the use of the data models supported by the database engines for which he is responsible.
R29: Identifying, in cooperation with the *project manager*, the content, format and location of the supporting documentation to be produced during the project within his developments scope.
R30: Designing the data model needed within his developments scope.

2) Cardinality: This role presents a mandatory nature and can even be performed simultaneously by several individuals in the same project, making each one responsible for a different line of development which, although possibly related to the other, has its own objectives and evolves independently of the others.

3) His Performance is Critical to: Ensure the goals achievement for each iteration.
Ensure the motivation of the *implementers* that he coordinates.
Protect the *implementers* he coordinates from internal and/or external pressures that may constrain their performance.
Detect non-feasible requirements or insufficiently described.

4) He Should Avoid to: Influence the commercial decisions of the *project manager*.
Oppose and/or reject to cooperate with the *software architect*.
Antagonize the *test manager* and/or not provide all the requested information.

5) At the Beginning of the Project He Should: Perform the followings responsibilities: R1, R6, R14, R19, R22, R25, R26, R27, R29 and R30.
Check if his holidays calendar is updated and if not, report it to the *project manager*.
Post, in a place defined by the organization, the following information: number and duration of the iterations to be performed (and its content) in agreement with the *project manager*; content, format and location of the internal documentation in agreement with the *process engineer*; definition of working procedures in agreement with the *process engineer*; reference to the data models used within his developments scope; identification of all external stakeholders connected with the project, their characterization and known contacts.

6) At the Beginning of Each Iteration He Should: Perform the followings responsibilities: R4, R10, *R18* and R21.
Assign tasks to the *implementers* that he coordinates.
Monitor the execution of the moving on to certification/production of the previous iteration produced artifacts.

7) During Each Iteration He Should: Perform the following responsibilities: *R1, R2,* R8, R9, R11, R12, R15, R16, R17, R20 and R28.
Promote weekly current status meetings (max. 30 min.) with the *implementers* that he coordinates.
Validate the feasibility of the established requirements for the system under his responsibility and, if this does not

happen, help to find an alternative considered viable and acceptable by the external entities.
Detect any new requirements arising from the implementation of the project which, after being properly documented, should be provided to the *project manager* (and if exists to the *system analyst*).
Detect the established requirements that are not sufficiently described and reporting them to the *project manager* (and if exists to the *system analyst*).
Detect possible requirements which have not been made explicit by external entities, and may be within their expectations or if they represent a business opportunity that could be exploited and inform the *project manager* (and if exists the *system analyst*).
Whenever justified, notify the *project manager* and the *process engineer* about the need to change the architecture within his developments scope and/or technical risks identified and update the documentation affected by it.
Prepare the required contents to execute the training plan offered to end users and to the several support teams (whether they are internal or external).

8) At the End of Each Iteration He Should: Perform the following responsibilities: *R5,* R13, *R22* and R23.
Perform the integration of the components implemented within his developments scope, in order to produce the required versions in each iteration, ensuring the availability to moving on to certification/production.
Ensure that all developed components are updated in the version control system.
Ensure that the *database designer* and *implementers* under his coordination have fulfilled their activities for closing the iteration.
Review if the project infrastructure requirements are maintained in the next iteration, and if not, notify the *project manager* about the necessary changes.
Evaluate, together with the *process engineer*, if the *implementers* under his coordination in the previous iteration are suitable and enough to his development scope in the next iteration.
Ensure that the existing data about the allocation of his working time are updated and available.
Review, with the *project manager*, the list of artifacts to produce in the next iteration, along with the technical characteristics of each, which shall include: applications: new features (described in free text, documents or diagrams), availability (online, outdoor installation, DVD, etc.), communication and image requirements (graphical interfaces, etc.); documents: languages, addressed topics, format (Word, Excel, PDF, etc.), communication and image requirements (using templates, etc.).
Check and notify the *project manager* about the need to make changes on holydays dates that match with the next iteration.
Ensure the availability of the necessary training and support material.

9) At the End of the Project He Should: Ensure that the *database designer* and the *implementers* under its coordination have fulfilled their project closing activities.
Communicate to the *process engineer* the assessment of the technical and behavioral skills of the *implementers* that he coordinated throughout the project.
Communicate to the *process engineer* the assessment of the components reused within his developments scope, identifying any correction or change to accomplish.
Collaborate, coordinated by the *project manager*, on the execution of a backup (at least in duplicate) of all relevant information (source code, artifacts, etc.) associated with his developments scope.
Communicate to the *process engineer* the assessment of the software development process used in the project, suggesting possible amendments or evolutions.

B. Implementer

The tasks associated to this role are generic by their definition, because they vary according to the requirements established for the components/systems to be developed. However, in general terms, it can be said that the commitment and professional pride that should guide the intervention of implementers will be prevalent for the fulfillment of the external commitments assumed by the organization and to obtain the desired quality levels for the artifacts implementers help to produce.

1) Responsible for: **R1:** Adopting best practices of software development.
R2: Performing with the utmost commitment and professional pride, the tasks assigned by the *integrator* that coordinates his work.
R3: Notifying the *integrator* if he wants to make a change in the holydays dates.
R4: Alerting as soon as possible his *integrator* when it is not possible to finish a task within the deadline.
R5: Reporting weekly to his *integrator* the time needed to complete his tasks.

2) Cardinality: This role has a mandatory nature and can be performed simultaneously by several individuals in the same project and divided by several lines of development.

3) His Performance is Critical to: Enable a possible corrective action, as early as possible, in situations of potential non-compliance with the objectives.
Create artifacts (applicational or not) with the quality level desired by the organization.

4) He Should Avoid: Not notify the respective integrator whenever he considers: not possessing the adequate knowledge to perform a task assigned to him; the deadline that was established to perform a given task is not enough.

5) At the Beginning of the Project He Should: Check if his holidays' calendar is updated and if not, report it to his *integrator*.
Post, in a place defined by the organization, the identification of all external stakeholders involved with the project, their characterization and contacts. (This is a generic task, for several roles, to ensure that everyone shares the information about the stakeholders with whom they have contact in the project).

6) At the Beginning of Each Iteration He Should: Request to the *integrator* the tasks allocation.

7) During Each Iteration He Should: Perform the following responsibilities: R1, R2, R3 and R4.

8) At the End of Each Iteration He Should: Ensure if the components source code that he is involved with is updated in the version control system.
Check and notify the *integrator* the need to make changes on the holydays dates that matches with the next iteration.
Ensure if the existing data about the allocation of his working time is updated and available.

9) At the End of the Project he Should: Communicate to the *integrator* the assessment of the components reused, identifying any correction or change to accomplish.
Communicate to the *process engineer* the assessment of the software development process used in the project, suggesting possible amendments or evolutions.

V. Case Study

A case study was developed to assess the *reduced model*. It involved seven development software teams. The software project developed by the teams was requested by a real customer that provided all the information about the organization and interacted directly with the teams.

The teams were constituted by second year students of the course 8604N5 Software System Development (SSD) from the undergraduate degree in Information Systems and Technology in University of Minho (the first University to offer in Portugal DEng, MSc and PhD degrees in Computing). The teams had between 13 and 17 people (1 team with 13, 3 teams with 14, 2 teams with 16 and 1 with 17). Each team receives an sequential identification number (Team 1, Team 2, .., Team 7) and the description of the customer problem. Two teams were randomly chosen to not adopting the RUP *reduced model* (we call these two teams the "Control Teams"), while the other five teams followed the guidelines established by the RUP *reduced model*, executing the phases of inception, elaboration and construction. The project lasted 3 months. The control teams did not follow any kind of guidelines for organizing themselves in term of roles/responsibilities/team organization.

The teams following RUP used the 8 roles proposed by the *reduced model*. Due to the complexity of the system, we have decided to instantiate two of the optional sub-roles referred in the Fig. 2: system analyst (that corresponds to a part of the responsibilities of the project manager) and software architect (that corresponds to a part of the responsibilities of the integrator). Team organization was as follows: 1 project manager, 1 or 2 system analysts, 1 or 2 integrators, 1 software architect, 1 project reviewer, 1 process engineer, 4 to 6 implementers (programmers), 1 system administrator, 1 test manager and 1 system tester.

The assessment of the *reduced model* was conducted by adopting the CMMI-DEV v1.2 ML2 reference model. With the exception of SAM (Supplier Agreement Management), all the other process areas had been assessed.

The diagnostic performed [32] within each of the teams adopted the following 5 steps: (1) a survey with 125 questions was developed based on generic and specific practices of CMMI-DEV v1.2 ML2; (2) the developed survey was assessed by 2 experts in SCAMPI model. The resulting suggestions were incorporated into the final version of the survey; (3) survey was answered by each of the project managers of the 7 teams; (4) each team element was characterize by mean of an online survey to collect information about age, sex, RUP role performed (except for the control teams), and the number of working hours. The survey response was 100%; (5) the RUP work products generated by each team were assessed in terms of their existence. This has allowed the validation of the data obtained from step 3 by each one of the project managers.

Table 1 shows the results obtained after the assessment: we present the percentage of accomplishment of specific practices for each process area. Although there is a significant difference between the various teams, the obtained results show that when the teams use the *reduced model* they are able to accomplish CMMI ML2 adequately.

The team average of the control teams is about 50%, while the average of the team averages of the teams that adopted the RUP *reduced model* is about 80% (two of these teams obtained averages in the scale of 90%). Interpreting these results we can conclude that the adoption of the *reduced model* allows an easier accomplishment of CMMI ML2.

VI. Conclusion

Throughout this work, we have shown that it is possible to configure the set of RUP roles in order to significantly reduce its size and thus maximize its use by an SME.

TABLE I. Case Study Results

CMMI-DEV ML2 assessed Process Areas	Identification of teams							Process Area average
	Team 1	Team 2	Team 3	Team 4	Control Team 5	Team 6	Control Team 7	
REQM- Requirements Management	64%	82%	93%	68%	50%	93%	36%	69%
PP - Project Planning	64%	77%	89%	68%	58%	86%	66%	74%
PMC - Project Monitoring and Control	76%	87%	97%	61%	39%	84%	58%	72%
MA - Measurement and Analysis	65%	91%	94%	50%	46%	91%	29%	67%
PPQA - Process and Product Quality Assurance	88%	81%	100%	54%	73%	92%	55%	78%
CM - Configuration Management	72%	81%	100%	72%	47%	91%	64%	75%
Team average	72%	83%	95%	62%	54%	90%	51%	

Consequently, we performed a reduction of the complexity embodied in the gathering of several RUP roles around a set of individuals who should be considered essential. As a consequence, we have described two models: (1) the *base model* presented in previous publications [4, 5]; and (2) the *reduced model*, a more pragmatic model, composed by eight distinct roles, that aims to allow an SME to effectively control the progress of their projects and avoid overlap and/or uncertainty of each individual scope of intervention.

Participants' performance in the software development process carried out in an SME is highly influenced by the limited range of human resources that usually accumulate a new role with other responsibilities in an ongoing project or in previous projects. So, we decided to describe the responsibilities of each role, to help each individual to know what is expected from him (by the exhaustive enumeration of his responsibilities) and also to identify the appropriate time to perform them (associating each of his tasks to a phase in the project). Additionally, we tried to reduce the margin of error, naming a specific individual to verify/approve the completion of an action item performed by another participant.

We have assessed the effectiveness of the reduced model, by using CMMI-DEV v1.2 ML2 to compare the maturity of teams that adopted the *reduced model* with the maturity of other teams that did not.

As future work, we will compare the maturity of teams that will adopt the *reduced model* with the maturity of other teams that will follow one agile methods approach, when considering CMMI-DEV v1.2 ML3 specific practices.

ACKNOWLEDGMENT

This work has been supported by FEDER through Programa Operacional Fatores de Competitividade – COMPETE and by Fundos Nacionais through FCT – Fundação para a Ciência e Tecnologia in the scope of the project: FCOMP-01-0124-FEDER-022674.

REFERENCES

[1] M. C. Paulk, et al., The capability maturity model: guidelines for improving the software process: Addison-Wesley, 1995.

[2] E. Commission. (2005, 2012-03-12). 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.: Addison-Wesley, 2003.

[4] P. Borges, et al., "Tailoring RUP to Small Software Development Teams," in SEAA 2011, pp. 306-309.

[5] P. Borges, et al., "Mapping RUP Roles to Small Software Development Teams," in SWQD 2012, pp. 59-70.

[6] C. E. de Barros Paes and C. M. Hirata, "RUP Extension for the Development of Secure Systems," in ITNG 2007, pp. 643-652.

[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] G. K. Hanssen, et al., "Using Rational Unified Process in an SME – A Case Study," in EuroSPI 2005, pp. 142-150.

[9] G. K. Hanssen, et al., "Tailoring RUP to a Defined Project Type: A Case Study," in PROFES 2005, pp. 209-228.

[10] H. Westerheim and G. K. Hanssen, "The introduction and use of a tailored unified process - a case study," in SEAA 2005, pp. 196-203.

[11] G. K. Hanssen, et al., "Tailoring and Introduction of the Rational Unified Process," in EuroSPI 2007, pp. 7-18.

[12] E. B. Pereira, et al., "A Systematic Approach to Process Tailoring," in ICSEM, 2007, pp. 71-78.

[13] M. Hirsch, "Making RUP agile," in OOPSLA 2002 Practitioners Reports.

[14] W. Hesse, "Dinosaur meets Archaeopteryx? or: Is there an alternative for Rational's Unified Process?," in SoSyM, vol. 2, pp. 240-247, 2003.

[15] B. Henderson-Sellers, et al., "A qualitative comparison of two processes for object-oriented software development," Information and Software Technology, vol. 43, pp. 705-724, 2001.

[16] B. Henderson-Sellers, et al., "Third generation OO processes: a critique of RUP and OPEN from a project management perspective," in APSEC 2000, pp. 428-435.

[17] L. V. Manzoni and R. T. Price, "Identifying extensions required by RUP to comply with CMM levels 2 and 3," in IEEE TSE, vol. 29, pp. 181-192, 2003.

[18] G. Chang, "Modifying RUP to comply with CMM levels 2 and 3," in ICISE 2010, pp. 1-5.

[19] J. Smith. (2000, 2012-03-12). Reaching CMM Levels 2 and 3 with the Rational Unified Process, http://www.uml.org.cn/SoftWareProcess/pdf/rupcmm.pdf

[20] B. Gallagher and L. Brownsword. (2001, 2011-02-10). The Rational Unified Process and the Capability Maturity Model – Integrated Systems/Software Engineering, http://www.sei.cmu.edu/library/assets/rup.pdf

[21] V. F. Del Maschi, et al., "Practical Experience in Customization of a Software Development Process for Small Companies Based on RUP Processes and MSF," in Management of Engineering and Technology, Portland International Center for, 2007, pp. 2440-2457.

[22] P. Abrahamsson, et al., "Agile Software Development Methods: Review and Analysis," Technical Research Centre of Finland, 2002.

[23] C. Alistair, Crystal clear a human-powered methodology for small teams: Addison-Wesley Professional, 2004.

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

[25] S. Ken, Agile Project Management With Scrum, Microsoft Press, 2004.

[26] B. Kent, Extreme programming explained: embrace change, Addison-Wesley, 2000.

[27] C. M. Ana Sofia, et al., "Mapping CMMI Project Management Process Areas to SCRUM Practices," in SEW 2007, pp. 13-22.

[28] S. W. Ambler. (2001, 2011-11-15). Agile Modeling and the Rational Unified Process (RUP). http://www.agilemodeling.com/essays/agileModelingRUP.htm

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

[30] R. S. Corporation. (2012-03-12). Roadmap: Agile Practices in RUP, http://sce.uhcl.edu/helm/RationalUnifiedProcess/tour/rm_xp2rup.htm.

[31] 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," in WER 2006, pp. 153-159.

[32] F. Mandjam, "Avaliação do impacto das práticas do CMMI no desempenho de equipas de desenvolvimento de software no ensino," Master Degree in Information Systems, University of Minho, 2011.