A Project management methodology for commercial software reengineering.

Ángel García-Crespo,

Ricardo Colomo-Palacios*,

Juan Miguel Gómez-Berbís

Computer Science Department, Universidad Carlos III de Madrid

Av. Universidad 30, Leganés, 28911, Madrid,

Spain

[angel.garcia, ricardo.colomo,juanmiguel.gomez]@uc3m.es

Marcos Ruano Mayoral

Av. Brasil, 17, 28020, Madrid,

Spain

marcos.ruano@egeoit.com

Abstract:

Changes in production, economic and technological environments are forcing organizations to update their Information Systems. In particular environments and circumstances, reengineering presents itself as a valid option to realize these updates. In organizations dedicated to the commercialization of software products, the reengineering process can be carried out iteratively. With the objective of supporting this process within computer software manufacturers, this paper presents a methodology for the project management of these types of activities, adopting an approach which focuses on continuous improvement of the process. This methodology is adaptable, firstly, to the requirements of organizations, secondly, to the characteristics of the software product. The focus of improvement of the reengineering process is performed in the model by means of continuous inspection and supply. The proposed methodology has been tested in the context of a European software vendor. The results of the implementation demonstrate a promising and feasible methodology to conduct commercial software reengineering efforts.

Keywords: Reengineering, Migration, Software Process Improvement, Project Management.

Reference to this paper should be made as follows: -----

Biographical notes:

Authors

Ángel García-Crespo is the Head of the SofLab Group at the Computer Science Department in the Universidad Carlos III de Madrid and the Head of the Institute for promotion of Innovation Pedro Juan de Lastanosa. He holds a PhD in Industrial Engineering from the Universidad Politécnica de Madrid (Award from the Instituto J.A. Artigas to the best thesis) and received an Executive MBA from the Instituto de Empresa. Professor García-Crespo has led and actively contributed to large European Projects of the FP V and VI, and also in many business cooperations. He is the author of more than a hundred publications in conferences, journals and books, both Spanish and international.

Ricardo Colomo-Palacios is an Assistant Professor at the Computer Science Department of the Universidad Carlos III de Madrid. He has been a Faculty Member of the Computer Science Department at Universidad Carlos III de Madrid since 2002. His research interests include Software Process Improvement, Software Project Management and Information Systems. He received his PhD in Computer Science from the Universidad Politécnica of Madrid (2005). He also holds a MBA from the Instituto de Empresa (2002). He has been working as software engineer, project manager and software engineering consultant in several companies including Spanish IT leader INDRA.

Juan Miguel Gomez-Berbís is an Assistant Professor at the Computer Science Department of the Universidad Carlos III de Madrid. He holds a PhD in Computer Science from the Digital Enterprise Research Institute (DERI) at the National University of Ireland, Galway and received his MSc in Telecommunications Engineering from the Universidad Politécnica de Madrid (UPM). He was involved in several EU FP V and VI research projects and was a member of the Semantic Web Services Initiative (SWSI). His research interests include semantic web, semantic web services, business process modeling, b2b integration and, recently, bioinformatics.

Marcos Ruano Mayoral is a consultant at EGEOIT, Spain. Formerly he was a Research Assistant of the Computer Science Department at Universidad Carlos III de Madrid. He holds a BSc in Computer Systems from Universidad de Valladolid and a MSc in Computer Science from Universidad Carlos III de Madrid. He has been involved in several research projects as information management engineer and software consultant.

1 Introduction

The spread of Information Systems in organizational environments in recent years has turned their development into a critical task for corporations. The software industry has become one of the main streams of development all around the world. In this scenario, organizations dedicated to the development and commercialization of software products represent an important volume of global economic data. Many of these organizations dedicated to the development of software packages face the problem of having to migrate or reengineer their products, adapting them to new technologies and functionalities, because changes in business processes are almost always linked to changes in systems and technology (Lientz & Rea, 2001b). In the case of software manufacturers, the requirements of modifying their technology imply the modification of the customer base, significantly multiplying the effects. The reengineering of software systems is widely recognized as one of the most significant challenges to be tackled by the software engineering community (Valenti, 2002).

The present initiative stems from the collaboration of Universidad Carlos III de Madrid in a project committed to the reengineering of a tool developed by a Spanish company. The main objective of this work is to combine and normalize the amount of tasks emerged from the reengineering requirements in software-intensive organizations. The proposed methodology represents a reference for the project management, the documentation elaboration and the evolutionary normalization of the tasks within an environment that allows the assessment and the continuous improvement of the involved processes, with a stress in the generalization of practices for reengineering other products developed by the company.

The remainder of this paper is organized as follows. Section 2 surveys the relevant literature. Section 3 describes the proposed methodology. Section 4 illustrates a use scenario. Finally, Section 5 outlines the main conclusions derived from the work and future work, concluding the paper.

2 Background. Reengineering and project management

The concept of Reengineering was described by Hammer and Champi (1993) as the the fundamental rethink and radical redesign of business processes to generate dramatic improvements in critical performance measures, such as cost, quality, service and speed. In practice, reengineering means to start over with a clean sheet of paper and rebuild the business better. Reengineering is only part of what is necessary in radical change of processes; it refers specifically to the design of the new process (Davenport, 1993). Figure 1 depicts the structure of this process.

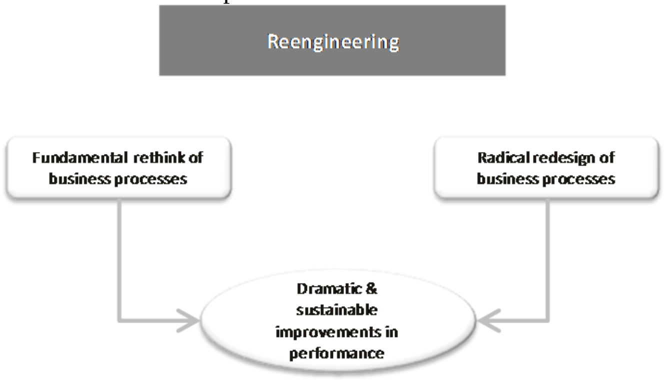
Reengineering
Fundamental rethink of business processes
Radical redesign of business processes
Dramatic & sustainable improvements in performance

Figure 1. Graphical reengineering concept

The importance of reengineering became particularly evident during the years following the publication of the works by Hammer and Champi. Undoubtedly, during this time, many reengineering efforts have met with problems or failure (Lientz & Rea, 2001). These authors state that over half of reengineering projects have failed (Lientz & Rea, 2001b). According to them the reasons can be found on the one hand, in the changing environment and the lack of measurement and control (Lientz & Rea, 2001), and on the other hand, because the business process side is too often ignored (Lientz & Rea, 2001b). These results encouraged Hammer and Champy to reinvigorate the topic in 2001. They reintroduce the goal of making major gains in reducing “waste” in the organization. They suggest that we reexamine every single process and rebuild businesses (Hammer & Champy, 2001). According to Attaran (2003), business process reengineering is going through its second wave, in which technologies like web services and knowledge management or e-learning play a crucial role. Finally, in the field of methodologies that support the business process reengineering process, the work by Stoica, Chawat and Shin (2004) states a revision of these tools which support reengineering, as well as a historical revision of its evolution.

The application of the concept of reengineering in the software field began to gain importance towards the end of the 1980s (Arnold, 1992). These efforts were conducted in order to reduce maintenance expense and improve software flexibility (Premerlani & Blaha, 1994). Further simpler and more general definitions of software reengineering also emerged in the literature. For example, experts have stated that software reengineering may be viewed as any activity that either improves the understanding of a software or else improves the software itself (Arnold, 1993). Chikofsky and Cross (1990) defined software reengineering as the examination and the alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form. Bearing in mind Arnold’s definition, the activities which comprise the process of software reengineering may be divided into two types according to Kullbach and Winter (1999). The first kind of activities are concerned with understanding, such as source code retrieval, browsing, or measuring. The second kind of activities aims at evolutionary aspects like redocumentation, restructuring and remodularization.

The role of software reengineering is vital, as it decreases complexity, increases quality and better equips future business environments (Behling, Behling & Sousa, 1996). Other authors argue that software reengineering is far more effective in terms of return of investment than another alternative of continuous improvement methods, such as maintenance (Deek et al., 2005). For the combating of what some authors have termed software aging symptoms (Visaggio, 2001) or to radically adapt the functionalities and technology of a software product, software reengineering presents a valid option.

It has been considered of such importance that since its initial application in productive environments, it has been regarded as an engineering problem (Feiler, 1993) that includes most aspects of traditional software development with additional constraints. As a consequence, since the middle of the 1990s many initiatives have emerged for software reengineering planification (Sneed, 1995), the establishment of a life cycle (Manzella & Mutafelija, 1992), taxonomies (Chikofsky & Cross, 1990) and economic aspects (Sneed, 1991). It was not until the millennium decade that efforts were focused on placing reengineering in an organisational environment, taking into account all the types of constraints which may be faced by a process by such characteristics. In this decade there are efforts conducted to improve legacy software on embedded systems (Sakai et al, 2004), develop software reengineering workbenches (Terekhov, 2004), tools (Cremer et al, 2002), documentation generation (Antoniol et al, 2002) or describe case studies in several technologies (Zou & Kontogianis, 2005) or within companies such as Inforsys (Mehta & Mehta, 2005).

....

There are many research efforts analyzing the success factors of software reengineering projects. In one of the most significant works, Teng *et al,* (1998) states the software engineering projects critical factors. In this study, authors cite the use of formal methodologies as one of the leading success factors. Perhaps having this latter critical factor in mind, the Software Engineering Institute of the Carnegie Mellon University, Bergey, O'Brien & Smith (2001) identified a model for Software Migration Planning, which has been applied in the U.S Department of Defense. The methodology is based on the management of the migration effort as the critical factor. Besides the mentioned model, a number of other efforts exist to define a process for the reengineering of software systems from a planification perspective (Sneed, 1995), such as iterative generation of software and specifications (Bianchi et al, 2003) or incremental processes (Brodie & Stonebraker, 1995).Undoubtedly, these latter models are more focused on the development and implementation aspects of software than those relating to the management of the project and the coverage of organizational needs.

Taking as a basis the characteristics of the model developed by Bergey, O'Brien & Smith (2001) and latter revisions by same authors (Bergey et al, 2002), the project team of Universidad Carlos III de Madrid has built a new software reengineering methodology. It is adapted to the characteristics of reengineering of commercial software packets, and additionally, is a methodology focused on continuous improvement to support the software reengineering processes of an organization.

3 A proposal for a software reengineering project management methodology

According to the specific requirements stated before, this article presents a software reengineering methodology built over two complementary tools closely related to the construction of information systems and software process. Firstly, ESA (European Space Agency) (ESA, 1991) software development standards PSS-05-0 has been adapted. These development standards will provide the guidelines for the code generation processes and will be the basis for the definition of the tasks related to the generation and analysis of the different solutions for a specific migration project.

The PSS-05-0 standard describes the processes involved in the complete life cycle of a single software project from its inception to the retirement of the software. The standard is divided into two areas, namely the Production Process, which has six phases and the Managing Process, which counts on four principal phases. The Production Process is composed by:

- User Requirements (UR) Definition Phase
- Software Requirements (SR) Definition Phase
- Architectural Design (AD) Phase
- Detailed Design (DD) Phase
- Transfer (TR) Phase
- Operations and Maintenance (OM) Phase

Additionally, the Managing Process is composed by four principal phases:

- Software Project Management
- Software Configuration Management
- Software Verification and Validation
- Software Quality Assurance

Fundamentally, the ESA specifications encourage the development of a formalized, knowledge-aware, cross-cultural set of guidelines to face software projects.

Secondly, we base our work in Software Engineering Institute's Software Migration Plan proposed by Bergey, O'Brien & Smith (2001). This recommendation defines a set of phases and tasks suitable for every reengineering process, and represents a relevant reference for the development of this kind of projects due to its origin and contents. The migration guidelines presented in the recommendation are complemented by a guide of migration practices extracted from the work by Bergey, Smith & Weiderman (1999) and developed at SEI too. With the objective that the methodology incorporates the Project manager vision, the models have been complemented with the proposal by Dey (1999), which proposes a Process reengineering for effective implementation of projects.

The resulting methodology is conceived as a set of activities separated from the rest of the business activities. Using a separate process for the reengineering of software with respect to other processes such as maintenance or development represents an established tendency in production environments (Jalote, 2002), in this case, once more, the Indian company Inforsys. In the environment in which the application of the methodology is considered, that is, the companies which develop commercial software, the ability to rely on a separate process for reengineering is an advantage, given that the elements for improvement are specifically applied, and by means of the communication of results and improved practices, spread through the entire organization.

After performing a process of analysis and definition of the solution, a methodology for the reengineering process developed has been established. The approach that we have outlined takes an active and dynamic view of migration planning based on different processes, which is a continuous effort that begins with the first increment of the migration plan. Thus the dynamics of our proposal, based on the works by is depicted in Figure 2.

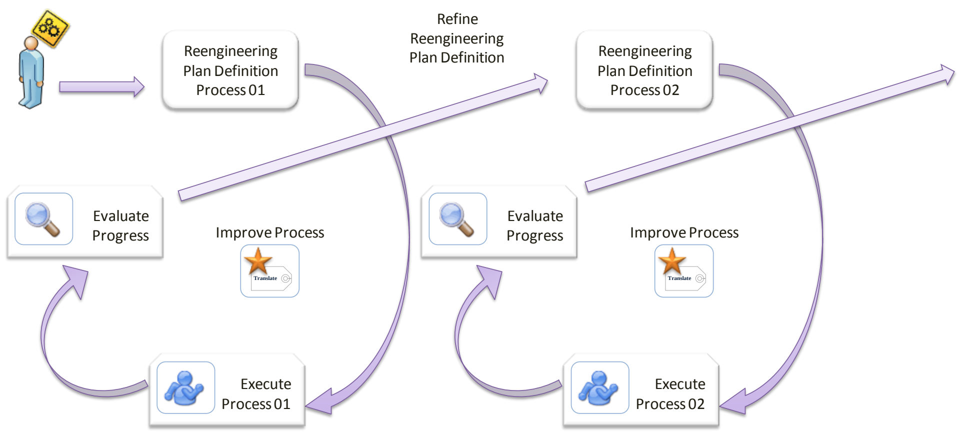

Figure 2. Dynamics of proposed reengineering project management methodology

The first increment of the plan deals with primary issues such as architecture and technology problems. Further iterations make the plan actionable and manageable. Each iteration or process provides more elaboration in the form of detailed process definition

....

the scope relevant to the active process. At any point in time, the reengineering plan can be viewed as the addition of the mini-plans of action corresponding to project processes accomplished. This will contain the focus area goal of the process, tasks & deliverables and resources with roles and responsibilities. Within a process, an iterative set of cycles are planned in order to provide an incremental development approach to the process. This incremental development approach benefits from the state of the art software engineering methodologies i.e. Barry Boehm’s spiral model (Boehm, 1988) or SCRUM (Schwaber & Beedle, 2002).

Thus, the proposed methodology lies on two iterative structures. Firstly, the one conformed by the sum of the processes which can be seen as a way to approach the whole problem in stages in order to adopt a *Divide et vinces* approach. A process can be seen as a deliverable of the new system, namely a version. Secondly, given a certain process execution, a set of cycles are proposed to implement iterative incremental software development. In this approach, instead of delivering a monolithic system after a long development time, smaller releases are implemented sequentially. According to Greer and Ruhe (2004), the benefits of this approach are twofold. First, requirements can be prioritized. Second, it means that customers receive part of the system early on and so are more likely to support the system and to provide feedback on it.

From the point of view of cost estimation, resource allocation and other project management fundamentals, the dynamics of proposed methodology are designed to bring the benefits of incremental construction method combined with software project management tools proposed by PSS-05-0. According requirement priorization, thus, though is a crucial project activity (Karlsson et al, 2007), proposed methodology is open to efforts like Karlsson, et al. (1998), Wiegers, (1999), EVOLVE by Greer and Ruhe (2004) or Quantitative WinWin (Ruhe et al., 2002). Requirements priorization must be made, firstly to draw project processes and within a process to set the cycles that rules the software construction.

A more in-depth structure of proposed methodology can be found in Figure 3.

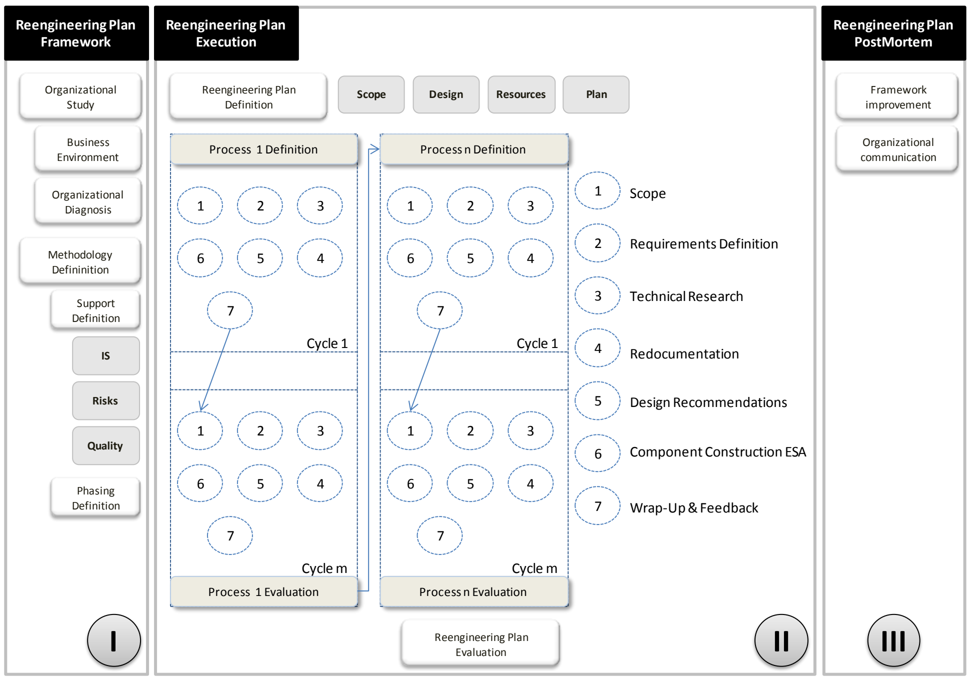

Figure 3. Software reengineering project management methodology

The proposed methodology is fundamentally based on two premises. The first one is the need of a methodology for the improvement of reengineering processes not only in the particular migration project in which it is deployed but in the whole of the organization. The second one is the will of obtaining a flexible methodology, without a stiff structure that hinders its adaptation to the specific scenario of the organizations where it is deployed.

I. Reengineering Plan Framework. It is a stage prior to the Reengineering Plan that is aimed at the design of the framework that will guide the process as a whole. This stage is the aggregation of the following phases:

I.1. Organizational Study. An analysis of the different organizational elements involved in the reengineering process is performed. Some of the aspects covered by this analysis are the current circumstances of the organization and the target system, the market characteristics and the involvement of the different organizational levels.

I.1.1. Business Environment. It is needed to carry up a detailed study including economic environmental factors, political environmental factors, social environmental factors and physical environmental factors.

I.1.2. Organizational Diagnosis. In this phase psychological and cultural climate are benchmarked. It is assessed by the perception of people with respect to variables which influence the motivation, work behaviour, technological status, financial performance, image, customer satisfaction and motivation of employees for excellence and as individuals and groups in the organisation (Dey, 1999). Additionally, it is reviewed financial performance, market image and growth trend indicate the characteristics of the organisation. These characteristics provide the context for the diagnosis of conditions in the organisation.

I.2. Methodology Definition. The main outcome of this phase is the definition of the reengineering methodology that best fits to organizational needs having in mind the conclusions extracted from Organizational Study and the different methodologies available at the market.

I.2.1. Support Definition. The objective of this phase is to establish and define the support requirements for the execution of the methodology in the target organization. These requirements are considered to be essential to the reengineering process and are totally independent of the tasks and activities performed in each process and cycle. Support Definition must include several issues that must be addressed. Firstly, Information Systems planning for effective project management including both automated and non- automated Information Systems. Secondly, project risk management including risk factors, effects and risk responses are taking into account. And thirdly, a quality assurance plan for that covers all the process.

I.2.2. Phasing Definition. This phase includes a definition of the different phases that structure the methodology. Those phases are established at this point since they are the basis of the Reengineering Plan Execution.

II. Reengineering Plan Execution. During the execution stage the actual migration tasks are performed. The suggested Reengineering Plan Execution is based on the Reengineering Plan Framework applied in the company context, which is detailed in the next section. The Reengineering Plan Execution stage defines a set of phases that are detailed next:

II.1. Reengineering Plan Definition. In this phase the characteristics of the migration to be tackled are defined. These characteristics include criteria relative to the number of reengineering processes to be performed, the scope of each of the processes, resources allocation and task plan. This plan is continuously updated on the basis of the information given in every cycle wrap-up and every process evaluation. It changes the criteria to select requirements and allocate resources.

II.2. Reengineering Process Definition. Each of the processes that make the plan up is an element that is tackled as an individual project for which scope and dependencies must be established. According to the Reengineering Plan Process, an arbitrary number of Reengineering Plan Processes can be instantiated to fulfil reengineering requirements in a phased approach. Each Reengineering Plan Process can be constructed by means of the following activities and tasks:

II.2.1. Process Plan. The Process Plan establishes the overall characteristics of the process including resources, plans (work breakdown structure), cycle descriptions, controls, success indicators and the products to be generated in each of the phases. This plan is based on the requirements priorization criteria of the process,that eventually, changes over time. The plan must answer following questions: What needs to be done? Who is going to do it? How will it be done? and How do we make sure that it is done satisfactorily?

II.2.2. Process Cycle. A cycle is an activity that integrates a set of tasks aimed at the reengineering of a planned and specific element or set of elements. Each cycle comprises the execution of a combination of the following tasks:

II.2.2.1. Cycle Scope. This task specifies the cycle in a formal way, stating the cycle's objectives, a detailed plan of resources and time, the organization of the rest of the tasks of the cycle and an in depth definition of the products obtained from those tasks.

II.2.2.2. Requirements Definition. The Requirements Definition task is focused on the elicitation and codification of the reengineering requirements concerning the defined scope.

II.2.2.3. Technical Research. Technical Research is aimed at increasing the knowledge acquired by the development team about the possible solutions to be adopted and their implications. An adequate execution of this task will guarantee a collection of reengineering recommendations strongly based on the reality concerning the product to be reengineered.

II.2.2.4. Redocumentation. Provided that, after the Technical Research task, the existing documentation in the field of study is found to be poor or inadequate, it is planned to allocate the necessary resources for its re-elaboration.

II.2.2.5. Reengineering Recommendations. From the requirements obtained in II.2.2.2 and the Technical Research the reengineering recommendations for each of those requirements will be specified.

II.2.2.6. Reengineering Component Construction. During this task the element established at the Cycle Scope is built. The development of this component will meet the previously elicited requirements and will apply the recommendations obtained in task II.2.2.5. The development process to be performed in this task will follow the guidelines of the PSS-05-0 standard for software development provided by ESA.

II.2.2.7. Wrap up & Feedback. This task sums up the whole cycle and documents it with the objective of being a feedback for the next cycles of the process.

The default organization of these tasks has a sequential nature and within a regular cycle all the tasks from II.2.2.1 to II.2.2.7 should be performed. However, it is possible to adapt this organization to the specific needs of a given cycle. This way, some tasks could be parallelized, i.e. II.2.2.3 and II.2.2.4, and even some of them could be discarded, but having in mind that tasks II.2.2.1 and II.2.2.7 are compulsory since they define the limits of each cycle.

II.2.3. Process Assessment. This task is aimed at the evaluation of the developed reengineering process, determining its completion and its contribution to new eventual reengineering processes.

II.3. Reengineering Plan Evaluation. Once all the Reengineering Plan Processes have concluded, the quality of both the product and the reengineering process is evaluated. Additional evaluation is performed over the results of the reengineering, whose real process ends in this phase.

III. Reengineering Plan Postmortem. The aim of this stage is to provide as much feedback as possible to the reengineering methodology to introduce improvements in both its definition and quality. A schema of continuous improvement is adopted to enrich the model with calibrations and customizations.It is proposed that the postmortem phase serves to carry out communication of the results in the operating environment of the company, and additionally, to extend some of the characteristics which have been improved in the process to contribute to the maturity of the software process of the organization in which it has been implemented.

....

The major improvements compared to the approach proposed by Software Migration Plan proposed by Bergey, O'Brien & Smith (2001) are shown below:

- Built in integration with commercial software development methodology.
- Continuous improvement approach.
- Commercial software development focus.
- Requirements priorization concerns.

4 Case study

The VV organization (fictitious name) is a corporation founded in the early 1990s. VV is present in 10 countries and its customers belong to more than 80 countries. In 2007 earned a 3,7M€ turnover that represents a 60% increase. R&D is one of the pillars of the company, thus VV invest a 20% of its revenue in that area.

The company most relevant product is ToM (fictitious name) with over 1,200 licenses sold all around the world. ToM is a solution for administrative management of organizations that range from companies of all sizes to government organizations. The development of ToM involves almost 100 people from R&D, Technology and QA departments working for the inclusion of new functionalities, the adaptation to changing environments and the customization for specific clients of the solution out-of-the-box. The development process is supported by an in-house developed methodology inspired in the European Space Agency Standards for software development projects.

VV objective is that ToM retains all of the functionalities developed for the tool, in its "Out of the Box" versions as well as in its customised versions developed by the customers. Without a doubt, the architecture of the tool demanded by the customer should have the capability to be modified in order to adapt to new technologies for the creation of new user interfaces. Given this environment, the project accomplished by the collaboration of the university and the company has as objective the provision of a methodological framework, and continuous improvement of the reengineering process. This latter requirement stems from the need to implement the methodology developed not only in the context of the ToM tool, but also in the remaining product range of the company. On the other hand, it should be stated that one of the reasons for applying the reengineering process is because of the scarce documentation which accompanies the ToM tool, developed during the technology boom without the application of rigorous standards.

As an initial test of the preliminary acceptance of the proposal, a pilot project for the deployment of the methodology was performed. The project team was divided into two groups with the same competential capabilities. Each of the groups, which were similar in composition and competence, was assigned a complete reengineering cycle with equivalent size and complexity (ten software requirements to cope and about 20 KLDC to review per group). Each group was comprised of five members, a team manager and four technicians. One of the groups adopted the proposed methodology while the other followed the own internal methodology of the company. After finalizing the pilot project several structured interviews were conducted in order to inspect team member opinions. The comments of the practitioners were categorical. The ones that did not use the new methodology stated that they felt lost while performing their tasks since they were not aware of the real progress of each task and they did not have any way to clearly define the phases, the processes and the requirements of their project.

Data were extracted relating to the work of both groups concerning three distinct aspects. In the first place, the documentation generated for support to the process was measured both new documentation and re-documentation, and secondly, the team managers were required to provide data regarding the rework task of the reengineering cycle. Lastly, the deviation in the work plans of both groups was measured. The deviation is measured in % effort planned. Table 1 shows main figures of both teams.

Table 1. Testing projects main figures

	New methodology	Old methodology
Rework rate	6%	10%
Plan Deviation	4%	14%
New documentation produced in words	36,000	26,500
Redocumentation in words	10,200	8,300

Data showed that the performance of the project which has adopted the new method is considerably higher. The Rework rate (one of the evils of software engineering) is slightly smaller, which affects a minor deviation from the plan. These data, which often involve some minor benefits with regard to the production of documentation, however, are based on an increased production of both new and revisited documentation.

With the objective of complementing this quantitative vision with a qualitative perspective, two aspects were examined with respect to the work developed. A group of three experts (comprised of technicians from the company) which had not taken part in the process examined the product developed, including software and documentation. Subsequent to the exam, a personal interview was held with each technician to request their opinion regarding two aspects of the projects analyzed. In the first place, the technicians were requested their opinion concerning the quality and value of the documentation generated, and secondly, about the applicability of the conclusions of the said documentation which the team managers had drawn up as part of their management duties. The three managers coincided in affirming that the documentation generated by the team who executed the methodology was of very high quality in two of the cases, and high quality in one case. All confirmed that the quality of the documentation of the team which used the methodology was superior to that generated by the other team of the project. In relation to the conclusions, the results are even more encouraging. The focus on continuous improvement which the methodology adopts was assigned an excellent valuation in one of the two cases, and very good in the other. All of the judges affirmed that the conclusions have a superior applicability to the applicability guaranteed by conventional methodology.

After the pilot phase, the methodology was applied to the reengineering of one of the core components of ToM. The project started with an initial process in which, after the inspection of 500,000 lines of code of the legacy components, 101 reengineering requirements where elicited in 350 versions. Those requirements where transformed in 102 reengineering recommendations with 241 versions. The reengineering recommendations were used to build 40 new classes that amount 8,000 lines of non-auto generated lines of code. Now, VV is performing another Migration Process adopting the methodology presented currently with good results. The model is now enriched with customizations aimed at the continuous improvement approach method and is integrated in its own management process. The new version of ToM is planned to be launched by mid 2010.

5 Conclusions and Future Work

This work has presented a methodology for the management of software reengineering projects of commercial software products. The design of the methodology was realized based on relevant contributions from the literature using two combined approaches. First of all, we focused on the process continuous improvement and secondly, pondered about the conjunction of several state of the art methods. From the point of view of the deployment of the proposed software reengineering methodology in a real production environment, two conclusions can be extracted. The first one is the improvement in the communication of the objectives of the reengineering process, its phases and products among the reengineering team. This conclusion stems from the qualitative feedback received from the resources involved in the process a stated in the previous section. The second conclusion is the applicability of the solution to reengineering environments. The methodology has proven to be flexible and adaptable to the environment where it was deployed, enabling the meeting of reengineering requirements.

Future works should be centered on extending the methodology from multiple viewpoints. In the first place, given that the methodology has been designed for the reengineering of commercial software products, the implementation phase of the system has not been addressed. Therefore, it is aimed as a first proposal to extend the methodology to develop its applicability in non-commercial legacy software engineering environments. In the second place, it is proposed that the present methodology is tested in two scenarios, that is, commercial software and legacy software, to establish the differences between both approaches, and customize the system based on the specific characteristics of each. Thirdly, it is suggested to carry out works which deal with the concept and application of refactoring in this type of projects. Lastly, the extension of the methodology is suggested to integrate different methodologies of the development of software combined within the reengineering process.

References

Antoniol, G., Canfora, G., DeLucia, A. & Merlo, E. (2002). Recovering Traceability Links between Code and Documentation, *IEEE Transactions on Software Engineering*, 28(10), 970-983.

Arnold, R. S. (1992). Software reengineering: a quick history. *Communications of the ACM*, 37(5), 13-14.

Arnold. R. S. (1993). A Roadmap Guide to Software Reengineering Technology. In Arnold, R.S. (Ed.) *Software Reengineering*. Los Alamitos, CA: IEEE Computer Society Press.

Attaran, M. (2003). Exploring the relationship between information technology and business process reengineering. *Information & Management, 41*(5), 585-596.

Behling, R., Behling, C. & Sousa, K. (1996). Software Re-engineering: Concepts and Methodology. *Industrial Management & Data Systems*, 96(6), 3-10.

Bergey, J., O'Brien, L. & Smith, D. (2002). *DoD Software Migration Planning*. Technical Note CMU/SEI-2001-TN-012. Available online http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tn012.pdf

Bergey, J., O'Brien, L. & Smith, D. (2003). *Application of an Iterative Approach to DoD Software Migration Planning, An*. Technical Note CMU/SEI-2002-TN-027. Available online http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tn027.pdf

Bergey, J., Smith, D. & Weiderman, N. (1999). *DoD Legacy System Migration Guidelines*. Technical Report CMU/SEI-99-TN-013. Available online http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tn013.pdf

Bianchi, A., Calvano, D., Marengo, V. & Visaggio, G. (2003). Iterative Reengineering of Legacy Systems. *IEEE Transactions on Software Engineering*, 29(3), 225-241.

Boehm, B.W. (1988). A Spiral Model of Software Development and Enhancement. *Computer, 21*(5), 61-72.

Brodie, M. & Stonebraker, M. (1995). *Migrating Legacy Systems - The Incremental Approach*, San Francisco, CA: Morgan Kaufmann Publishers.

Chikofsky, E.J. & Cros, J.H. (1990). Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, 7(1), 13-17.

Cremer, K., Marburger, A. &Westfechtel, B. (2002). Graph-based tools for re-engineering. *Journal of Software Maintenance and Evolution: Research and Practice,* 14 (4), 257-292.

Davenport, T.H. (1993). *Process innovation: reengineering work through information technology*. Boston, MA: Harvard Business Press.

Deek, F.P., McHugh, J.A. & Eljabiri, O.M. (2005). *Strategic Software Engineering: An Interdisciplinary Approach*. Boston, MA: Auerbach.

Dey, P.K. (1999). Process re-engineering for effective implementation of projects. *International Journal of Project Management, 17*(3), 147-159.

European Space Agency (1991). ESA Software Engineering Standards ESA PSS-05-0. Issue 2. Available online ftp://ftp.estec.esa.nl/pub/wm/wme/bssc/PSS050.pdf

Feiler, P. (1993). Reengineering: An Engineering Problem (CMU/SEI-93-SR-005, ADA267117). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, July 1993. Available online: http://www.sei.cmu.edu/publications/documents/93.reports/93.sr.005.html.

Greer, D. & Ruhe, G. (2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, *46*(4), 243-253.

Hammer, M. & Champy, J. (1993). *Reengineering the Corporation.* New York: Harper Collins.

Hammer, M. & Champy, J. (2001). *Reengineering the Corporation: A Manifesto for Business Revolution*. New York: Harper Collins.

Jalote, P. (2002). *Software project management in practice*. Boston, MA: Addison-Wesley.

Karlsson, J., Wohlin, C. & Regnell, B. (1998). An Evaluation of Methods for Prioritising Software Requirements. *Information and Software Technology, 39* (14/15), 939-947.

Karlsson, L., Thelin, T., Regnell, B., Berander, P. & Wohlin, K. (2007). Pair-wise comparisons versus planning game partitioning--experiments on requirements prioritisation techniques. *Empirical Software Engineering, 12*(1), 3-33.

Kullbach, B. & Winter, A. (1999). Querying as an enabling technology in software reengineering. *Proceedings of the Third European Conference on Software Maintenance and Reengineering*, pp. 42-50.

Lientz, B.P. & Rea, K.P. (2001). *Project Management for the 21st Century*, Third Edition. Oxford, UK: Butterworth-Heinemann.

....

Lientz, B.P. & Rea, K.P. (2001b). *Breakthrough Technology Project Management*, Second Edition. Oxford, UK: Butterworth-Heinemann.

Manzella, J. & Mutafelija, B. (1992). Concept of re-engineering life-cycle. *Proceedings of the Second International Conference on Systems Integration*, pp. 566-571.

Mehta, N. & Mehta, A. (2005). Inforsys Technologies Limited. In Jennex, M. (Ed.) *Case Studies in Knowledge Management*. Hersley, PA: Idea Group.

Premerlani, W. & Blaha, M. (1994). An approach for reverse engineering of relational databases. *Communications of the ACM*, 37(5), 42-49.

Ruhe, G., Eberlein, A. & Pfal, D. (2002). Quantitative WinWin: a new method for decision support in requirements negotiation. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, pp 159-166

Sakai, M., Kubota, M., Okita, M., Matsumoto, K.I. & Torii, K. (2004). A new environment for improving legacy software on embedded systems. *Systems and Computers in Japan*, 35(9), 81-91.

Schwaber, K. & Beedle, M. (2002). *Agile Software Development with Scrum*. Upper Saddle River, NJ: Prentice Hall.

Sneed, H.M. (1991). Economics of software re-engineering. *Journal of Software Maintenance: Research and Practice*, 3(3), 163-182.

Sneed, H.M. (1995). Planning the Reengineering of Legacy Systems. *IEEE Software*, 12(1), 24-34.

Stoica, M., Chawat, N. & Shin, N. (2004). An Investigation of the Methodologies of Business Process Reengineering. *Information Systems Education Journal, 2* (11).

Teng, J., Jeong, S. & Grover, V. (1998). Profiling successful reengineering projects. *Communications of the ACM*, 41 (6), 96-102.

Terekhov, A.A. (2004). Dealing with Architectural Issues: a Case Study. *ACM SIGSOFT Software Engineering Notes*, 29(2), 11-11.

Valenti, S. (2002). *Successful Software Reengineering*. Hersley: PA: Idea Group.

Visaggio, G. (2001). Ageing of a data-intensive legacy system: symptoms and remedies. *Journal of Software Maintenance: Research and Practice*, 13(5), 281-308.

Wiegers, K. (1999). *Software requirements*. Redmond, WA: Microsoft Press.

Zou, Y. & Kontogianis, K. (2005). Reengineering legacy systems towards web environments. In Khan, K. and Zheng, Y. (Eds.) *Managing Corporate Information Systems Evolution and Maintenance*. Hersley: PA: Idea Group.