Improving Distributed Software Development in Small and Medium Enterprises

Miguel Jiménez[1,*], Aurora Vizcaíno[2] and Mario Piattini[2]

[1]*Alhambra-Eidos, Paseo de la Innovación 1, 02006, Albacete, Spain*

[2]*University of Castilla-La Mancha, Alarcos Research Group, Institute of Information Technologies & Systems, Escuela Superior de Informática, Paseo de la Universidad 4, 13071, Ciudad Real, Spain*

Abstract: One of the current tendencies of software enterprises is that of making greater development efforts in more attractive zones by decentralizing their production units. Small and Medium Enterprises (SMEs) are a very important cog in the application of Distributed Software Development (DSD). The software industries of many countries are made up mainly of small and medium software enterprises which in many cases employ this approach by taking advantage of the greater availability of human resources in decentralized zones at a lower cost. However, this leads to certain disadvantages which are mainly due to the distance that separates the teams. Coordination and communication become more difficult, thus affecting productivity and product quality. Efficient Software Engineering practices which are adapted to SME characteristics are therefore necessary. In this paper, we review the main challenges and proposals relating to DSD which may be useful in SME environments, with the principal purpose of providing a set of methods and techniques that can be applied in a generic environment.

Keywords: Distributed Software Development, nearshoring, offshoring, Small and Medium Enterprises.

1. INTRODUCTION

One of the current trends of the software industry is that of relocating its production units throughout distributed sites, principally in order to take advantage of the greater availability of a skilled workforce and also to take into consideration political and economical factors [1], thus allowing organizations to increase their market area by producing software for remote clients. The main objective of this consists of optimizing resources in order to develop higher quality software and minimizing costs.

Distributed Software Development (DSD) allows team members to be located at various remote sites during the software lifecycle, thus forming a network of virtual teams that work on the same projects. These teams might be members of the same organization or might require the collaboration or outsourcing of different organizations. Although this phenomenon came into being during the 90`s, only during the last decade has its strategic importance been recognized [2], and related studies are quite recent [3].

Organizations which apply DSD commonly use iterative approaches in contrast to traditional waterfall or sequential methods, as these become more difficult to use consistently when teams are geographically distributed [4].

In these environments problems [5] caused mainly by distance appear [6], which must be confronted by concentrating on the specific context of each organization. Traditional face-to-face meetings are, therefore, no longer common, communication is less fluid than in co-localized development groups and interaction between members requires the use of technology to facilitate communication and coordination [7], thus enabling organizations to abstract themselves from geographical distance and minimizing the negative impact on development productivity and software quality.

This situation influences the way in which software is defined, built, tested and delivered to customers, thus affecting the development methodology applied [5], which must be adapted to achieve higher levels of productivity through new technologies, processes and methods [8].

In this paper we deal with this subject from the point of view of Small and Medium Enterprises (SMEs). There are several definitions of an SME which concentrate on indicators such as the number of employees and financial criteria. The European Commission [9] describes an SME as an independent firm which employs less than 250 employees. According to this definition, 99.2% of software development companies in the world are SMEs [10], and a large number of initiatives related to the improvement of their processes exist [11], which will be dealt with in our study.

SMEs are different to large enterprises with regard to the application of DSD in the complexity of their structure and organization. Large enterprises usually have more problems which are caused by the teams' diversity and the size of the projects.

SMEs commonly use quality models such as CMM and CMMI [12] (promoted by the Software Engineering Institute) and quality standards, such as ISO 9001:2000 [13], which follow organizational structures that automate parts of

*Address correspondence to this author at the Alhambra-Eidos, Paseo de la Innovación 1, 02006, Albacete, Spain; E-mail: Miguel.Jimenez@a-e.es

1874-107X/10

their software development, decentralizing their production units and promoting the reusability of architectures, knowledge and components.

The aim of this work is firstly to identify the best procedures, models and strategies dealt with in literature in order to improve the application of DSD in SMEs, and secondly to propose a set of methods and guidelines which could be part of a methodology that would allow SMEs to conduct a DSD process in an efficient manner, thus increasing their efficiency and productivity. This paper is, therefore, organized as follows: Section 2 describes the main challenges of DSD in SMEs and presents the most useful proposals found in literature. Section 3 presents a set of the techniques proposed based on the authors' experience and the analysis of literature with the aim of improving DSD in SMEs. Finally, Section 4 provides concluding remarks and key success factors.

2. CHALLENGES AND PROPOSALS ORIENTED TOWARDS DSD

In this section we summarize the main challenges and proposed improvements identified in studies related to DSD which may be useful in defining a methodology for SMEs. This will allow us not only to discover related works but also to enumerate the best techniques and methods for the subjects addressed. The systematic review of the literature applied can be found in [14].

2.1. Communication

During the software lifecycle, team members exchange a large amount of information using different tools and different formats, usually without following any communication standards, and thus encountering misunderstandings, high response times and security problems. When the members are located in different countries (termed as Global Software Development (GSD)), then other problems arise such as misinterpretation, since the members are using a language which is not their mother tongue to communicate. However, in this work we intend to focus on DSD in general without taking into account whether members are spread throughout different countries, since SMEs seldom have subfactories in other countries. However, we wish to highlight that all the issues dealt with in this paper are applicable to GSD.

All these drawbacks cause a decrease in communication frequency that directly affects productivity and development quality. In order to decrease these effects, a methodology for DSD must be supported by collaborative tools, so as to avoid ambiguity and face-to-face meetings without comprising the quality of the results [15]. It is therefore recommendable to institutionalize collaboration processes such as those examined by Thissen *et al.* [16], in which conference calls and e-mails play an important role.

The use of translation processes, and codification guidelines is useful in the case of pronounced cultural differences [17], but this will not be taken into account for SMEs in this work. However, different levels of understanding of the problem domain may exist, along with the different levels of knowledge, skills and training of the team members. It is therefore necessary to introduce user-friendly tools, and integrate collaborative tools to improve knowledge integration [18].

Requirements elicitation is one of the processes which is most frequently involved in communication, signifying that it must be clearly defined and modelled in order to make it easily understood, and promoting its traceability with use cases. With the aim of reducing communication problems, Aranda *et al.* [19] propose a technique to improve requirements elicitation by selecting a suite of groupware tools and techniques from the field of cognitive psychology. The team members' communication skills are also a critical success factor, and the separation of responsibilities in the team must therefore take into account their psychological skills.

2.2. Configuration Management

As the degree of distribution of the team grows, coordination and synchronization become more complex, and traceability is a critical factor. Source control systems must support access through Internet to avoid conflicts, and must therefore confront its unreliable and insecure nature and the higher response times, and supporting role-based access control.

Several studies that propose new tools to reduce these drawbacks exist. One is CHIME [20], an Internet and Intranet based application which allows users to be placed in a 3D virtual world representing the software system. This system allows users to interact with project artefacts by "walking around" the virtual world and collaborating with other users through a feasible architecture that provides an overview of the ongoing activities in the project. Another related tool is FASTDash [21], which uses a spatial representation of the shared code base and highlights team members' current activities, allowing a developer to rapidly determine which team members have source files checked out, which files are in use, and what methods and classes are currently being edited, providing immediate awareness of potentially conflictive situations.

2.3. Knowledge Management

The team members' experiences, methods, decisions, and skills must be accumulated during the software lifecycle through effective information-sharing mechanisms, so that team members can access this experience, thus increasing productivity by avoiding redundant work.

SMEs must facilitate knowledge sharing by maintaining a product/process repository by linking the content from sources, such as e-mails or chats, and sharing metadata information among several kinds of tools. Zhuge [22] presents an approach which is based on a knowledge repository in which the information associated with each project is stored by using Internet-based communication tools, thus enabling new team members to become quickly experienced.

2.4. Quality Management

The impact of problems in DSD projects can be magnified, and it is usually more difficult to recover from them than in collocated projects. Organizations should introduce new quality assurance models and measures to obtain information which can be adapted to the distributed scenarios, thus ensuring that the requirements reflect the customers' needs. One of the most frequently recommended practices in

SMEs through which to increase the code quality is the automation of code inspections and the application of coding standards [23].

Quality must not only be limited to software products but also to development processes, which greatly influence product quality. Software evaluation also plays a key role in product quality, which usually involves a large number of stakeholders who need face-to-face evaluation meetings, and appropriate collaborative tools are therefore needed [15].

With this aim in mind, the capability model eSCM-SP [24] considers the factors that influence software quality management systems from a cultural and organizational perspective. This model gathers the best practices, and is quite similar to other capability-assessment models such as CMMI, Bootstrap or SPICE and the SQM-CODE model.

Our study of the relative literature has led us to observe a lack of empirical studies that permit the enumeration of reliable measures. More papers related to tests in distributed environments, which are directly related to software quality, are also necessary. However, we also discovered interesting proposals such as the *interdependence measure* [25], which permits the measurement of the degree of work dispersion among sites to be determined by looking up the locations of all the individuals. F. Lanubile *et al.* [7] similarly propose metrics associated with products and processes oriented towards software defects such as: discovery effort, reported defects, defects density, fixed defects or unfixed defects.

2.5. Risk Management

In distributed environments, risk management also includes new issues apart from those connected with collocated environments. DSD development includes new problems related to coordination, problem resolution, evolving requirements, knowledge sharing and risk identification [8]. Software defects become more frequent due to the added complexity which is, in most cases, related to communication problems and a lack of group awareness. Rules and guidelines with which to organize the teams and their interactions become necessary. Teams must be continuously controlled in order to detect problems and take corrective actions, and the use of suitable measures is an important key factor.

In order to minimize these problems, F. Lanubile *et al.* [7] define a process in which roles, guidelines, forms and templates are specified. They also describe a Web-based tool that adopts a reengineered inspection process in order to minimize synchronous activities and coordination problems, thus supporting geographically dispersed teams.

The WOOM [26] methodology also provides measures and facilitates decision making, taking into account both the risks during various lifecycle phases and mitigation plans.

2.6. Project Management

Project management also becomes more difficult as a result of high organizational complexity, and the new challenges in scheduling, task assignment and cost estimation, which are influenced by volatile requirements, tasks interdependencies, changing specifications and the lack of informal communication [27].

Madachy [28] deals with economic estimation, presenting a set of cost models that are valid for SMEs, which takes into account the teams' various environmental characteristics, localized labour categories, calendars, compensation rates, and currencies for costing.

The use of mature processes becomes a key success factor, as does applying incremental integrations and frequent deliveries, it being recommendable to follow informing and monitoring practices [29].

SMEs require the automation of the development process through an adaptable tool to manage tasks and metrics through customizable reports managed by a central server, and ensuring that the process is applied in compliance with a predefined standard. The SoftFab infrastructure [30] is a related approach which enables projects to automate the building and test process, and which manages all the tasks remotely through a control centre.

The availability of real-time information about the members' participation also helps managers in decision making. Gousios *et al.* [31] proposes a model for evaluating developers' contributions by combining traditional metrics with data mined from software repositories to extract contribution indicators.

2.7. Process Support

Software development implies that all team members must follow a significant number of steps. DSD environments often use multiple sub-processes that are distributed across different locations. Such systems must offer support for distributed execution, data access and the use of collaborative tools [32, 33]. Modern management workflow systems use the Web as a means to provide access to the workflow engine [34]. However, most of these systems are based on the client-server architecture [35], with the problems that this entails for communication, and the dependence on the correct functioning of the server.

Processes should reflect the direct responsibilities and dependencies between tasks, providing notifications based on roles to inform those concerned about the changes and new tasks assigned, thus avoiding overloading team members with too much information. Problems derived from process evolution and mobility appear within the context of DSD for SMEs. Furthermore, distributed environments usually involve a large network of heterogeneous, autonomous and distributed models and process engines. Many studies are directed towards the integration of heterogeneous processes that work with different models and engine support [36].

One environment that supports the creation and exploitation of software process models is known as PSEE (Process-centered Software Engineering Environment) [37]. A PSEE oriented towards distributed environments must be designed to be adaptable to changes during enactment, taking into account the following requirements:

- Enable incomplete processes. A PSEE must facilitate the incremental editing of models, allowing their refinement during the projects' lifecycle.

- Be adaptable to specific projects and their changing needs, caused both by changes in requirements and by new technological advances and tools.
- Changing a process should be a simple and secure operation. Managers must control the overall development process, improving it during enactment and minimizing any factors that may decrease productivity.
- The system must be fault tolerant, and the probability of errors must be minimized.
- Provide support to coordination, cooperation and monitoring [38], along with information about activities through automatic notifications based on roles.
- Provide availability and reliability through the replication of servers.

The processes are commonly modelled by using a Process Modeling Language (PML) [39], which can assist SMEs in different manners within the field of DSD:

- Process understanding: A PML can be used to accurately represent the structure and organization of a process [40]. Several environments also exist, such as Spearmint [41], Rational Method Composer [42] and Eclipse Process Framework Composer [43], which are able to generate structured process workflow-oriented guidelines from the process definition.
- Training and education: A precise description of the process may be useful to introduce the team to the procedures and operations carried out by the organization.
- Distributed process design: A PML can be used to design new processes, describing their structure and organization. One example which is useful in DSD is that of the Spearmint environment, which supports extensive capabilities for multi-view modelling and analysis.
- Process simulation and optimization: Processes can be simulated to evaluate potential problems and identify bottlenecks and areas for improvement. S. Setamanit *et al.* [44] describe a hybrid computer simulation model of software development processes to study alternative ways in which to configure DSD projects in order to confront communication problems, control and coordination problems, process management and time and cultural differences.
- Interoperability: PMLs promote the interoperability between the different systems which take part in a distributed development.

Wang *et al.* [45] present a framework for assessing the degree to which PMLs are suitable to represent processes in distributed environments, dealing with five main areas: distribution, autonomy, diversity, collaboration and flexibility.

The Model Driven Development (MDD) approach is currently emerging in this field, providing reusability, maintainability, interoperability and adaptability through different languages and platforms. Model Driven Architecture (MDA) [46] is the most frequently adopted MDD standard, and provides concepts of separation in individual models and transformation techniques. An example of this can be found in InterDOC [47], an approach to enable the authoring process when interoperability among different collaborative applications is necessary.

Numerous environments related to process support in distributed environments appear in literature, such as MILOS [48], GENESIS [49], PROSYT [50], OPERA [36] and Oz [51].

2.8. Coordination

In distributed environments, coordination becomes more difficult owing to problems caused by communication, lack of group awareness and complexity, which influence the way in which the work must be structured and managed [52]. This additional complexity requires the participation of more people, which causes delays [53]. Virtual teams are also prone to be less productive as a result of their feelings of isolation and indifference.

The figure of a local coordinator for each site becomes essential, as is the use of collaborative tools to permit monitoring activities and managing dependencies. More progress reports, project reviews, conference calls and regular meetings to take corrective action are therefore necessary, thus minimizing interdependencies between tasks assigned to different virtual teams [54]. Ariadne [55] is a tool which analyzes software projects for dependencies and helps to discover coordination problems through a visual environment.

Literature deals with models based on empirical data that allow organizations to calculate the impact of coordination efficiency and its effects on productivity. Setamanit *et al.* [44] describe a simulation model to study different ways in which to configure global software development processes.

Developers need to have as much information as possible available and to have a full knowledge of the full status of the project and its past history, which will allow them to become more involved in the project and to make better decisions. In this respect, YooHoo [56] is an awareness system that helps developers to keep up to date with code changes. Information about these changes is filtered on the basis of the developer's interests, and notifications are provided in a flexible manner.

Members usually have problems finding the right person and timely information, which may cause inefficiencies that result in misalignment, replanning, redesign and redevelopment. Augur [57] is a visualization tool which addresses these drawbacks, and supports DSD processes by creating visual representations of both software artefacts and software development activities, thus allowing developers to explore the relationships between them.

2.9. Collaboration

Distributed environments imply the necessity for collaboration between business analysts, customers, system engineers, architects and developers. It is common the use of multi-agent models that facilitate communication among distributed members. We can find an example in [58], which

allows identifying the required information of the activity and the best moment to interrupt other members and entering into collaboration.

The concurrent edition of models and processes requires synchronous interaction between dispersed architects and developers. This necessitates concurrency control in real time, thus enabling members to edit and discuss the same diagrams, and providing a means through which to easily capture and model complex concepts through virtual workspaces [23].

One approach is that of the SoftDock framework [59], which solves the issues related to software component modelling and their relationships by describing and sharing component model information, and ensuring their integrity. It permits developers to analyze, design and develop software from component models and transfer them by using an exchange format, which facilitates communication between team members.

One relatively extended method, owing to its accessibility and adaptability, consists of using wikis. Galaxy Wiki [60] is an on-line collaborative tool based on this concept which permits the existence of a collaborative authoring system for documentation and coordination purposes, and which allows developers to compile, execute and debug programs on wiki pages.

We should also mention collaborative code editors such as the Sangam system [61], an Eclipse plug-in which is oriented towards distributed pair programming and which allows distributed developers to share a workspace in order to see and edit the same code. Similarly, IMPROMPTU [21] is a framework for collaboration in multiple display environments, which allows users to share task information through displays via off-the-shelf applications.

3. METHODS AND GUIDELINES PROPOSED

In this section we propose a methodology oriented towards DSD in SME environments, taking into account the limited complexity and budget of these organizations which typically lead them to apply simplified methodologies, paying particular attention to their organizational structure. Not all of the activities proposed by the common standards (such as ISO/IEC 12207 [62]) are always suitable for these environments, which also apply lower levels of maturity in comparison to larger companies. Our proposal is made up of a set of methods and guidelines whose purpose is to improve productivity and guarantee the quality of the final product in compliance with the CMMI level 3 standard. Its definition has taken into account the information presented in the previous section, along with the authors' previous experience after using a traditional methodology.

We have also considered the use of an **iterative development lifecycle**, since cascade models are not recommendable in these environments.

3.1. Communication

On occasions, developers may need to contact other remote developers who are working on different parts of the software. However, it is not always possible to know which person to contact, so it is advisable to carry out communication through the **local sub-director** who must manage all the communications for that site and that project. The distribution of **organizational charts** [29] which identify the location of members must also be considered. This could be useful in locating members.

The use of **suggestions** [63] is also recommended. This concept is based on the idea of carrying out communication through well-structured templates that will guide the participants in the management of information to improve communication by reducing the number of interactions required. This method should be used in all formal communication between distributed members, storing the information generated in a shared repository, thus helping prevent duplicate conversations and improving the overall knowledge of the status of the project.

Furthermore, it is also necessary to foster informal communication, which will take place through the use of instant messaging programs and e-mail.

Useful Tools

- Synchronous traditional tools (such as video-conferences and chats)
- Asynchronous communication tools based on suggestions and traditional e-mails

3.2. Configuration Management

In DSD environments configuration management tools must work under a unified process through a centralized repository that stores all the software artefacts (documentation, source code, etc.) and which permits the definition of roles with different permissions and responsibilities.

The tool employed should be able to replicate the information in the different locations so that each virtual team can work from a local server which is regularly synchronized with the central server. This would improve local time access, thus avoiding the periods of inactivity caused by communication problems.

Some of the most valuable characteristics of a configuration management tool in DSD are:

- Possibility of deploying a proxy server at the remote locations to minimize the dependency of the central server.
- Possibility of defining check-in policies according to the organization's specific needs.
- Facilities for branching and merging the code.
- Shelving; this allows files on the server to be saved with pending changes that will not affect the current project, thus permitting them to be stored on the server without the existence of a compilable version, and making them available to distant users.
- Automatic builds; in order to compile the code and execute tests with a certain periodicity, or which are triggered by certain events.
- Automatic generation of reports.

A good product structure reduces the need for branches and consequently the risks of merging the different versions. It is therefore essential to start from a good analysis.

Finally, it is also necessary to establish an **auditory process** that allows the maturity of the project to be checked and that permits all the artefacts to come under a configuration management. This activity should be carried out jointly with the sub-directors of the virtual teams.

Deliverables	Useful Tools
- Audit reports	- Configuration management tool - Audity process automation tool

3.3. Knowledge Management

Distributed environments should make use of a system that facilitates collaborative knowledge management between remote sites. Many kind of **collaborative tools** are reported in literature, the most popular in this field being the wiki pages [8].

It is recommended that each project has a Web portal integrated with other tools involved in the software lifecycle, with customizable content in which members can share documents through a control versions system and manage schedules, meetings, calendars, etc., thus allowing access control based on roles and using collaborative spaces.

One member of each virtual team should assume the role of **document manager,** who is responsible for the local content in the portal. In some cases, the different documents will need different tools for their edition, it being desirable to use the fewest possible number of tools to facilitate ease of access to the information. The organization must institutionalize **document templates** to provide team members with a guide to help them to organize the information through the tools employed by the organization.

Useful Tools
- Collaborative tools (such as wiki pages and project Web portals)

3.4. Quality Management

In distributed environments, it is important to obtain information about the progress of the different teams and to ensure the existence of automated tools that will guide the development in compliance with the established standards. This issue is of great importance in CMMI which, through the area of "*Measurement and Analysis*", provides guidance about what organizations need to focus their improvement efforts on, and which is orientated towards the planning, monitoring, control and evaluation of software processes [64].

One of the most frequently employed strategies consists of the automation of **code inspections** through a tool that checks certain coding rules that are institutionalized within the organization.

Quantitative information about the project status and progress must be readily available, thus facilitating decision making. In our proposal, all information related to the software lifecycle is stored as suggestions, which allows all the team members to view and follow all the formal interactions.

The **global coordinator** of quality assurance is responsible for defining the quality management strategy. Each virtual team will have a **local quality manager** who is responsible for the fulfilment of the local strategy.

Each team member should complete an evaluation form with a certain frequency, which should be sent to the global coordinator through the local coordinators. The global coordinator will then review the evaluation forms to complete a final report on the problems, and this will be sent to the local coordinators who should take corrective actions according to this document.

Finally, the **metrics** used must be adjusted by taking into account the new factors introduced by DSD, adding variables such as team size, the percentage of reused code, size of the code or effort (person / hours), combined with other traditional metrics.

We recommend the use of the work dispersion variable, which is similar to the Herfindahl-Hirschman Index [65], to quantify the degree of distribution of the work, and which is defined as follows for the case of two sites:

Work dispersion $= 100^2 - (\%\ effort\ in\ the\ first\ factory)^2 - (\%\ effort\ in\ the\ second\ factory)^2$

Deliverables	Useful Tools
- Review evaluations - Problems report	- Code inspection automation tool - Metrics management tool

3.5. Risk Management

The Risk Plan must be developed by considering the risks arising from the new problems caused by DSD that directly affect productivity and the project budget. This also necessitates the adaptation of the Mitigation Plan. As in the case of requirements, here it is also necessary to maintain records of the changes. The information related to the teams' interaction must also be stored. Team members must be able to report problems encountered during development, which will speed up problem solving and will permit a list of problems to be maintained that will serve to create a realistic risks catalogue.

It is also recommendable to conduct **periodic surveys** that take up little time and allow developers to provide project managers with a vision of the progress and the problems encountered.

In response to the indications of CMMI, it is vital to keep track of the Risk Plan with a certain frequency (which could depend on the type of project), and the immediate response to the occurrence of risks during the project lifecycle must be ensured. It is therefore advisable to designate a local person responsible for monitoring at each distributed location, who must reach a consensus with the global person responsible for determining priorities and actions to be carried out.

Deliverables
- Risk list (including new risks derived from DSD) - Mitigation plan - Surveys concerning problems discovered - Risk Reports

3.6. Project Management

Project planning should minimize dependencies among the different sites to minimize accumulated delays. This decision is reflected in the **planning design distribution** document, which is elaborated during the final stage of analysis design, and should be based both on historical data and on the opinion of the local sub-directors. Generally, a higher coupling level between work units will require a greater coordination effort. Therefore the task distribution should consider the organization's structure and its possibilities for coordination and communication.

It is recommendable to take into account the fact that those activities that require extensive knowledge of the developed system, such as the development of installers or the generation of documentation, should be carried out at the location at which the highest percentage of time has been devoted to the development.

The **project planning** document should also indicate where each module will be developed. During this stage an **interaction plan** must be defined among participants, which should determine how to perform the interaction between virtual teams in the case of specific needs.

The planning should consider frequent deliveries, which will increase the developers' motivation and will permit the definition of several control points in order to make early decisions by concentrating on project goals, software quality or development costs.

It is therefore recommendable to use a **cost estimation** method that takes into account the number of distributed teams, their size, the time at the different locations, the costs associated with distance, etc. [28].

According to the CMMI recommendations, it is vital to specify the frequency with which tracking is conducted, in which key aspects of development must be reviewed to ensure the project's consistency. A formal document that determines how to perform this review must be institutionalized.

Once the project has finished, a set of data that could be used to analyze the final results should be collected. Concretely, it would be of interest to have an **estimation database** containing the following information:

- Brief description of project and technology employed.
- Structure of project tasks, their description and people involved.
- Development distribution degree and sites involved.
- Number and complexity of the modules.
- Variation between the estimated duration and budget versus actual.
- List of incidents or problems not covered by the Risk Plan.

This information must be used to estimate the planning of subsequent projects through methods based on the detection of analogies.

Deliverables	Useful Tools
- Project planning - Interaction plan - Cost estimation	- Cost estimation tool based on detection of analogies

3.7. Process Support

The process support must provide mechanisms with which to carry out process improvement. The CMMI *Process Improvement* area provides guidelines to facilitate a better development in future projects using the experience accumulated from previous experiences. This task requires the establishment of a criterion with which to periodically evaluate the processes followed by the organization in order to identify improvements through a **process assessment report**.

Taking into account this document and the projects evaluation results, a committee should agree on a **process improvement planning** document. The implementation of this improvement should include the collaborative edition of the processes, through a tool that will permit the participation of distributed members.

After implementing the process improvements, the changes applied should be reflected in the **process guide**. This document may be accessed by any member of the team through the project portal.

The implementation of changes in the processes requires the use of a **version control** that permits a return to a previous state in the case of inconsistencies. It is also necessary to know the process that each project is using, and its version.

All the changes implemented in the process must be tested, which requires the elaboration of a **process test plan** to ensure that the objectives of the improvement plan are met without errors. With the aim of solving problems which have gone unnoticed during the tests and of fulfilling the changing needs of the projects, it should be possible to make changes in processes even during enactment.

The possibility of having distributed processes that run on different local servers is not generally considered to be necessary in SME environments.

Deliverables	Useful Tools
- Process assessment report. - Process improvement planning. - Process guide. - Process test plan.	- Collaborative edition of processes - Automatic generation of process guides - Process management with version control - Distributed process engines

3.8. Collaboration

In relation with collaboration, we have focused on the software design and analysis phase. Documentation generated in this phase is critical for the virtual teams, and the analysts that defines the architecture should be as reduced as far possible, since having too many analysts might be coun-

terproductive, especially if they cannot have face-to-face meetings.

A tendency currently exists which involves explicitly describing the decisions made during the design phase and explaining the **reasoning** behind the making of such decisions [66]. The main objective of this consists of providing developers with all the information related to the context of the problem, which might help them to understand the design reasons. A rational design must include [67]:

- The reasoning behind a design reason.
- The different alternatives taken into account.
- The drawbacks of each alternative.
- Justification for each decision.
- The reasons that led to this decision.

One recommendable method in DSD consists of recording the realization of the analysis by using a video and other means, and storing the reasoning in a knowledge base so that the information can be consulted by any team member. The rational design benefits DSD in the following manners:

- Verification: it can be used to verify whether the design decisions correspond with designers' and users' expectations.
- Evaluation: it can evaluate the various alternatives that were discussed in the design.
- Maintenance: it helps to evaluate the changes.
- Reutilization: it helps to develop new requirements from previous reasoning.
- Learning: it facilitates the learning of the system to new users.
- Decision making: the information stored may be useful in the decision making process in subsequent projects, thus avoiding the repetition of mistakes.
- Documentation: it can be used to document the design process.

If a project member is unable to attend a meeting, s/he could use the documentation generated to discover the decisions that were made. Various tools focused on rational design exist, such as bCisive (http://bcisive.austhink.com/) or Compendium (http://compendiuminstitute.org).

Once the engineering design phase is completed, it is necessary to discuss the **planning and tasks division**, establishing a separation of the work between the different sites. This decision must be agreed with the local sub-director of each site, identifying the most appropriate team, and attempting to minimize the communication requirements between sites, taking into account the fact that the critical tasks must be assigned to the most experienced members.

3.9. Coordination

The first step in the development process must consists of designating a **global director** to coordinate the project and a **local sub-director** for each distributed site that takes part in the project lifecycle.

The local sub-director is in charge of coordinating the work at his/her site, and is responsible for assigning resources to the team and monitoring its activities, taking local corrective actions when necessary. All communication with other sites should be carried out through the local sub-directors, the global director being the person the responsible for the global coordination and the success of the project.

The next step consists of assigning the requirements that team members must fulfil, defining their associated responsibilities and tasks. Potential team members will be selected by considering these requirements in order to strengthen performance and their interaction.

CMMI establishes that a document with the **team structure** should be institutionalized. The size of the teams should be as reduced as far possible in order to facilitate collaboration. In addition, it will be necessary to evaluate the team periodically and modify its structure in order to adapt to the project's changing necessities.

In some cases the project's requirements may make it necessary to designate an **integration team**, which should be made up of at least one member of each virtual team with the aim of reducing communication times. In projects with a high degree of distribution, a higher frequency of integrations would be desirable.

It would also be recommendable to employ the methodology presented by Aranda *et al.* [19], in which a set of guidelines are proposed to help to determine the problems that might appear during the development, concentrating on team members' **cognitive factors** and the project's characteristics.

This analysis should be performed through the use of the Felder-Silverman test [68], which gathers information about the psychological behaviour of each person, which in turn serves as an indicator of how an individual perceives, interacts and reacts to the environment. This information classifies learning styles, thus classifying the individual as sensing/intuitive, visual/verbal and sequential/global. This information is extremely useful when carrying out the assignation of tasks and in determining the tools and elicitation techniques that are more appropriate for each team.

Although this work does not consider cultural differences, it might also be of interest to make use of a glossary of terms for all the documents related to the project in order to avoid ambiguities and assure a common understanding.

Finally, **regular meetings** between virtual teams should be planned. If possible, a first meeting with all the members

Deliverables	Useful Tools
- UML models - Reasoning about decisions - Development division planning	- Rational design tools (such as bCisive or Compendium)

of the project is recommendable. This could take place at the location of the greatest number of members involved in the project in order to minimize travel costs. This meeting encourages a better understanding between and implication of the members, and will promote future informal communications. The technical specialist should then visit the virtual team regularly in order to monitor and audit the project and carry out demonstrations of the project.

Deliverables	Useful Tools
- Responsibilities assigned to each team - Structure of teams - Planning of meetings	- Groupware tools presented by Aranda *et al.* [19]

3.10. Requirements Elicitation

Requirements Engineering (RE) activities must promote the understandability of the requirements and establish their priorities. UML models [52, 69] must facilitate visual comprehension by avoiding ambiguities.

During the initial phase, the requirements must be defined through meetings with the customers, during which the presence of the local sub-directors of each virtual team is desirable if misunderstandings in subsequent phases are to be minimized.

The organization must institutionalize a set of **criteria for the requirements selection**, which will establish a basis for their acceptation with certain uniformity, concentrating on the organization's characteristics.

The **requirements specification** obtained must be contained in a document which is accessible to all the members. This document must be more detailed than in the case of a co-localized environment, and must indicate the motive, context and related decisions of every requirement, avoiding assumptions and misunderstandings through a **glossary of terms**. It is also necessary to state the reasons behind the rejection of those requirements that have been ruled out.

The initial creation of the document must be based on a predefined template which assures the generation of a well structured and detailed specification. During the development, this document must be reviewed periodically, and meetings with the members will be helpful to assure a common understanding.

One of the most critical aspects of RE in DSD environments is the exchange of information and the notification of changes to the team members [70]. Both the **automation of notifications** and the bidirectional traceability between requirements and design models and tests through links that permit the impact of changes to be discovered are consequently necessary. Historical information with regard to requirements must also be accessible to facilitate the discovery of the reasons behind certain decisions, and to assist the person responsible for any changes.

3.11. Software Testing

The high complexity of the systems which must be tested and the communication difficulties in DSD environments necessitate that the testing process begin as soon as possible to facilitate an early detection of errors. This task requires the use of specific tools, and their interoperability with other tools involved in the software lifecycle. Distributed sites should interact through a **local testing manager** at each factory, who is in charge of managing the local tests.

On several occasions, similar test cases are modelled at different levels of granularity. A set of guidelines to standardize the formal test modelling is therefore necessary. The use of Model-Based Testing (MBT) [71] is thus recommended. This method permits the automatic generation of efficient test procedures through the use of the system's requirements and functional specifications. MBT provides the following advantages:

- Planning is shorter, at a lower cost and is of higher quality.
- It improves communications between developers and test engineers.
- Early discovery of ambiguities during specification and design.
- Easy to update before changes in test requirements.
- Ability to manage software quality.

Various case studies concerning several kinds of models [72], and reports regarding the integration of MBTs into industrial environments [73] exist in literature.

Once the Test Plan has been developed, the **distribution of testing tasks** between the different sites must be planned, deciding which teams will be responsible for conducting each set of tests. This decision must be agreed with the local sub-directors of each site.

It is necessary to use **collaborative tools** that facilitate testing tasks and monitor them so that the analysts can be aware of the status of the project at any time. Formal communication between the team members involved in the testing must be carried out through **suggestions** to standardize the information of the tests.

Deliverables	Useful Tools
- Testing plan - Design division planning - Integration testing report	- MBT tools for the collaborative generation of test procedures

Deliverables	Useful Tools
- Requirements specification document - List of criteria for the requirements selection - Traceability matrix - Changes carried out and analysis of their impact	- Concurrent edition of UML models - Traceability and change management automation

4 CONCLUSIONS

In this paper we have provided a global vision of the challenges related to the DSD which is oriented towards SMEs, and the main proposals found in literature to tackle them. Moreover we propose various strategies that could form part of a DSD methodology. In these strategies we describe certain tools for and solutions to the different problems which, according to literature and our own experience, occur when SMEs begin to develop software in distributed settings. These guidelines and recommendations can therefore be used by SMEs to adapt a traditional software process model to a distributed model. These proposals attempt to be generic and extensible to SMEs, and are based on the experiences of a company that applies DSD.

Every organization has concrete needs which basically depend on its distribution characteristics, its activity and the tools it employs. These factors therefore cause this subject to be extremely wide-ranging, and lead to the necessity to adapt both the technical and organizational procedures, according to each organization's specific needs. However, we have attempted to present some proposals that could be applicable to a wide range of SMEs.

The application of agile methodologies based on incremental integration, frequent deliveries, and frequent reviews of problems to adjust the process becomes an important success factor, along with the application of maturity models such as CMM or CMMI, which provide a good basis through which to carry out adaptation towards DSD.

The tools employed by the organization must be adapted and integrated, thus assuring their interoperability and suitability for the application of the proposed methods.

The development process must be automated through a tool that provides an efficient communication mechanism between the members of the organization. The application of an appropriate PML and the use of environments such as Spearmint, Rational Method Composer or Eclipse Process Framework Composer for the model definition are essential to the generation of structured process guidelines which will facilitate process understandability and the training of human resources in the processes introduced.

Our immediate future work will include the empiric validation of the proposed methodology through the definition of a set of metrics that will allow us to quantify the improvement based on a set of study cases. In order to achieve this, we shall apply the Research-Action method [74] to our target company, which will be useful in that it will allow us to study the problems of this methodology and propose new solutions.

ACKNOWLEDGMENTS

We would like to acknowledge the assistance of the MELISA project (PAC08-0142-3315) which was financed by the "Junta de Comunidades de Castilla-La Mancha" of Spain, PEGASO project (TIN2009-13718-C02-01), financed by the "Ministerio de Ciencia e Innovación" of Spain and MEVALHE project (HITO-09-126) funded by Consejería de Educación y Ciencia, Junta de Comunidades de Castilla-La Mancha, co-funded by Fondos FEDER. This work is part of FABRUM project (PPT-430000-2008-63), financed by the "Ministerio de Ciencia e Innovación" of Spain and by Alhambra-Eidos (http://www.alhambra-eidos.es/).

REFERENCES

[1] W. Aspray, F. Mayadas, and M. Y. Vardi, "Globalization and offshoring of software. A report of the ACM job migration task force," New York 2006.

[2] R. Davison, "Offshoring information technology: sourcing and outsourcing to a global workforce," *Inf. Technol. Dev.*, vol. 13, no. 1, pp. 101-102, 2007.

[3] R. Prikladnicki, D. Damian, and J. L. N. Audy, "Patterns of evolution in the practice of distributed software development: quantitative results from a systematic review," In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE)* University of Bari, Italy, 2008.

[4] M. A. Cusumano, "Managing software development in globally distributed teams," *Commun. ACM*, vol. 51, no. 2, pp. 15-17, 2008.

[5] L. Layman, L. Williams, D. Damian, and H. Bures, "Essential communication practices for Extreme Programming in a global software development team.," *Inf. Softw. Technol.*, vol. 48, no. 9, pp. 781-794, 2006.

[6] S. Krishna, S. Sahay, and G. Walsham, "Managing cross-cultural issues in global software outsourcing," *Commun. ACM*, vol. 47, no. 4, pp. 62-66, 2004.

[7] D. Damian, F. Lanubile, and H. Oppenheimer, "Addressing the Challenges of Software Industry Globalization: The Workshop on Global Software Development," *ICSE*, pp. 793-794, 2003.

[8] R. Sangwan, M. Bass, N. Mullick, D. J. Paulish, and J. Kazmeier, *Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series)*, Auerbach Publications: Boston, MA, USA, 2006.

[9] OECD, "OECD small and medium enterprise outlook," Organisation for Economic Co-operation and Development, Paris, 2002.

[10] M. Laitinen, M. Fayad, and R. Ward, "Software engineering in the small," *IEEE Softw.*, vol. 17, no. 5, pp. 75-77, 2000.

[11] F. J. Pino, F. García, and M. Piattini, "Software process improvement in small and medium software enterprises: a systematic review," *Softw. Quality Control*, vol. 16, no. 2, pp. 237-261, 2008.

[12] CMMI, *Capability Maturity Model Integration (CMMI). Version 1.2*: Software Engineering Institute. Carnegie Mellon., 2006.

[13] J. Zavala-Ruiz, "Organizational Analysis of Small Software Organizations: Framework and Case Study," In: *Software Process Improvement for Small and Medium Enterprises: Techniques and Case Studies*, H. Oktaba, and M. Piattini, Eds. USA: IGI Global, pp. 1-41, 2008.

[14] M. Jiménez, and M. Piattini, "Problems and Solutions in Distributed Software Development: A systematic Review," In: *Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD)*, Zurich, pp. 107-125, 2008.

[15] M. A. Babar, B. Kitchenham, L. Zhu, I. Gorton, and R. Jeffery, "An empirical study of groupware support for distributed software architecture evaluation process," *J. Syst. Softw.*, vol. 79, no. 7, pp. 912-925, 2006.

[16] M. R. Thissen, J. M. Page, M. C. Bharathi, and T. L. Austin, *Communication tools for distributed software development teams*. ACM Press: St. Louis, Missouri, USA, 2007.

[17] J. M. Carey, "Creating global software: A conspectus and review," *Interact. Comput.*, vol. 9, no. 4, pp. 449-465, 1998.

[18] K. M. Babu, "Globalization and offshoring of software," *Ubiquity*, vol. 7, no. 43, pp. 2-2, 2006.

[19] G. N. Aranda, A. Vizcaíno, A. Cechich, and M. Piattini, "Strategies to recommend groupware tools according to virtual team characteristics," In: *Conference on Cognitive Informatics, 2008. ICCI 2008. 7th IEEE International* Stanford, CA, pp. 168-174, 2008.

[20] S. E. Dossick , and G. E. Kaiser, *CHIME: a metadata-based distributed software development environment*. Springer-Verlag: Toulouse France, 1999.

[21] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson, *FASTDash: a visual dashboard for fostering awareness in software teams*. ACM Press: San Jose, California, USA, 2007.

[22] H. Zhuge, "Knowledge flow management for distributed team software development," *Knowledge-Based Syst.*, vol. 15, no. 8, pp. 465-471, 2002.

[23] L. Al-Jadiri , and B. Bruegge, "Enabling offshore software testing: A case study," In: *Proceedings of the 11th IASTED International Conference Software Engineering and Applicattions*, Cambridge, MA, USA, 2007, pp. 282-289.

[24] B. B. Kerstin, V. Siakas, "Software outsourcing quality achieved by global virtual collaboration," *Softw. Process Improv. Pract.*, vol. 11, no. 3, pp. 319-328, 2006.

[25] J. D. Herbsleb, and D. Moitra, "Guest editor's introduction: Global software development," *IEEE Softw.*, vol. 18, no. 2, pp. 16-20, 2001.

[26] R. Kuni, and N. Bhushan, "IT application assessment model for global software development," In: *International Conference on Global Software Engineering (ICGSE'06)*, pp. 92-100, 2006.

[27] I. Richardson, V. Casey, D. Zage, and W. Zage, "Global Software Development – the Challenges," University of Limerick, Ball State University, SERC Technical Report 278 September 2005.

[28] R. J. Madachy, "Cost modeling of distributed team processes for global development and Software-Intensive Systems of Systems," *Softw. Process Improv. Pract.*, vol. 13, no. 1, pp. 51-61, 2008.

[29] M. Paasivaara, and C. Lassenius, "Collaboration practices in global inter-organizational software development projects," *Softw. Process Improv. Pract.*, vol. 8, no. 4, pp. 183-199, 2003.

[30] H. Spanjers, M. T. Huurne, B. Graaf, M. Lormans, D. Bendas, and R. v. Solingen, "Tool support for distributed software engineering," In: *International Conference on Global Software Engineering (ICGSE'06)*, pp. 187-198, 2006.

[31] G. Gousios, E. Kalliamvakou, and D. Spinellis, *Measuring developer contribution from software repository data*. Leipzig, Germany: ACM, 2008.

[32] Z. B.-S. Israel, and E. K. Gail, *A paradigm for decentralized process modeling and its realization in the Oz environment*. IEEE Computer Society Press: Sorrento, Italy, 1994,

[33] T. Pierre Fernand, *Modelling the Federation of Process Sensitive Engineering Environments: Basic Concepts and Perspectives*: Springer-Verlag, 1998.

[34] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, *Capability Maturity Model for Software, Version 1.1*, Pittsburgh: Software Engineering Institute, 1993.

[35] P. Garg, *Process-Centered Software Engineering Environments*, IEEE Computer Society Press: Los Alamitos, CA, USA, 1995.

[36] C. J. Hagen, *A Generic Kernel for Reliable Process Support*. Zurich, 1999.

[37] L. Osterweil, *Software processes are software too*. Monterey, IEEE Computer Society Press: California, United States, 1987.

[38] A. Fuggetta, *Software process: a roadmap*. Limerick, Ireland: ACM, 2000.

[39] K. Z. Zamli, "Process modeling languages: a literature review," *Malaysian J. Comput. Sci.*, vol. 14, no. 2, pp. 26-37, 2001.

[40] S. Bandinelli, A. Fuggetta, L. Lavazza, M. Loi, and G. P. Picco, "Modeling and improving an industrial software process," *IEEE Trans. Softw. Eng.*, vol. 21, no. 5, pp. 440-454, 1995.

[41] U. Becker-Kornstaedt, L. Scott, and J. Zettel, *Process engineering with Spearmint/EPG*. Limerick, Ireland: ACM, 2000.

[42] IBM, Rational Method Composer – Rational unified process, version 7.1, 2006.

[43] P. Haumer, Eclipse Process Framework Composer: Part 1: Key Concepts. Technical Report, IBM Rational Software, 2006.

[44] S.-o. Setamanit, W. Wakeland, and D. Raffo, "Using simulation to evaluate global software development task allocation strategies," *Softw. Process Improv. Pract.*, vol. 12, no. 5, pp. 491-503, 2007.

[45] A. I. Wang, R. Conradi, and C. Thuv, "A Framework for Evaluating Process Modelling Languages for Distributed Environments", in Peter Kokol (Ed.): P*roc. IASTED International Conference on Software Engineering and Applications (SEA 2005)*, Phoenix, Arizona, USA, ACTA Press, pp. 168-176, November 14-16, 2005.

[46] Object Management Group. MDA guide version 1.0.1. OMG Document Number omg/2003-06-01, June 2003.

[47] R. S. P. Maciel, C. G. Ferraz, and N. S. Rosa, "An MDA domain specific architecture to provide interoperability among collaborative environments," In: *19º Brazilian Symposium on Software Engineering*, 2005.

[48] S. Goldmann, J. Münch, and H. Holz, "A meta-model for distributed software development," In: *IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1999. (WET ICE '99) Proceedings.*, Stanford, CA, USA, pp. 48-53, 1999.

[49] L. Aversano, A. D. Lucia, M. Gaeta, P. Ritrovato, S. Stefanucci, and M. L. Villani, "Managing coordination and cooperation in distributed software processes: the GENESIS environment," *Softw. Process Improv. Pract.*, vol. 9, no. 4, pp. 239-263, 2004.

[50] G. Cugola, and C. Ghezzi, "Design and Implementation of PROSYT: A Distributed Process Support System," In: *IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Stanford, CA, USA, pp. 32-39, 1999.

[51] I. Z. Ben-Shaul, and G. E. Kaiser, "Federating Process-Centered Environments: The Oz Experience," *Automated Softw. Eng.*, vol. 5, no. 1, pp. 97-132, 1998.

[52] B. Berenbach, and M. Gall, "Toward a unified model for requirements engineering," In: *Proceedings of the IEEE international conference on Global Software Engineering*, pp. 237-238, 2006.

[53] J. D. Herbsleb, and D. Moitra, "Global software development," *IEEE Softw.*, vol. 18, no. 2, pp. 16-20, 2001.

[54] P. Ovaska, M. Rossi, and P. Marttiin, "Architecture as a coordination tool in multi-site software development," *Softw. Process Improv. Pract.*, vol. 8, no. 4, pp. 233-247, 2003.

[55] C. R. d. Souza, S. Quirk, E. Trainer, and D. F. Redmiles, *Supporting collaborative software development through the visualization of socio-technical dependencies*. Sanibel Island, Florida, USA, ACM, 2007.

[56] R. Holmes, and R. J. Walker, *Promoting developer-specific awareness*. Leipzig, Germany: ACM, 2008.

[57] J. Froehlich, and P. Dourish, "Unifying artifacts and Aactivities in a visual tool for distributed software development teams," In: *Proceedings of the 26th International Conference on Software Engineering*, Washington, DC, USA, pp. 387-396, 2004.

[58] R. R. Palacio, A. L. Morán, V. M. González, and A. Vizcaíno, "Providing support for starting collaboration in distributed software development: A multi-agent approach," In: *2009 World Congress on Computer Science and Information Engineering (CSIE 2009)*, Los Angeles/Anaheim, USA, 2009.

[59] J. Suzuki, and Y. Yamamoto, "SoftDock: A distributed collaborative platform for model-based software development," In: *10th International Workshop on Database & Expert Systems Applications* 1999.

[60] W. Xiao, C. Chi, and M. Yang, *On-line collaborative software development via wiki*. Montreal, Quebec, Canada: ACM, 2007.

[61] C.-W. Ho, S. Raha, E. Gehringer, and L. Williams, "Sangam: a distributed pair programming plug-in for Eclipse," In: *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange*, Vancouver, British Columbia, Canada, pp. 73-77, 2004.

[62] W. Lloyd, M. B. Rosson, and J. Arthur, "Effectiveness of elicitation techniques in distributed requirements engineering," In: *10th Anniversary IEEE Joint International Conference on Requirements Engineering, RE'02*, Essen, Germany, pp. 311-318, 2002.

[63] K. Narayanaswamy , and N. M. Goldman, "A flexible framework for cooperative distributed software development," *J. Syst. Softw.*, vol. 16, no. 2, pp. 97-105, 1991.

[64] D. Goldenson, J. Jarzombek, and T. Rout, "Measurement and analysis in capability maturity model integration models and software process improvement," *CROSSTALK J. Defense Softw. Eng.*, vol. 6, no. 7, pp. 20-24, 2003.

[65] F. M. Scherer , and D. Ross, *Industrial Market Structure and Economic Performance*: Houghton Mifflin Company, 1990.

[66] A. P. Jarczyk, P. Löffler, and F. M. S. III, "Design rationale for software engineering: a survey," In: *25th Hawaii International Conference on System Sciences*, HI, USA, pp. 577-586, 1992.

[67] J. Lee, "Design Rationale Systems: Understanding the Issues," *IEEE Expert Intellig. Syst. Appl.*, vol. 12, no. 3, pp. 78-85, 1997.

[68] R. M. Felder , and L. K. Silverman, "Learning and Teaching Styles in Engineering Education," *Eng. Educ.*, vol. 78, no. 7, pp. 674-681, 1988.

[69] B. Berenbach, "The automated extraction of requirements from UML models," In: *Proceedings of the 11th IEEE International Conference on Requirements Engineering*, pp. 287-288, 2003.

[70] M. Heindl, and S. Biffl, "Risk management with enhanced tracing of requirements rationale in highly distributed projects," In:

Proceedings of the 2006 international workshop on Global software development for the practitioner, Shanghai, China, pp. 20-26, 2006.

[71] I. K. El-Far, and J. A. Whittaker, "Model-based Software Testing," In: *Encyclopedia on Software Engineering*, J. J. Marciniak, Ed., Wiley, pp. 1-22, 2001.

[72] SoftwareTech, *Software Acquisition Gold Practice Model-based Testing (Webpage)*, 2008.

[73] H. Robinson, "Obstacles and opportunities for model-based testing in an industrial software environment," In: *Proc. 1st European Conference on Model Driven Software Engineering*, Germany, pp. 118-127, 2003.

[74] E. W. Eisner, "On the differences between scientific and artistic approaches to qualitative research," *Educ. Res.*, vol. 10, no. 4, pp. 5-9, 1981.

Received: July 01, 2009 Revised: July 30, 2009 Accepted: May 03, 2010