GLOBAL SOFTWARE DEVELOPMENT: CURRENT CHALLENGES AND SOLUTIONS

Juho Eskeli[1] and Jon Maurolagoitia[2]
[1] *Flexible Software Development Team – VTT, Oulu, Finland*
[2] *Product & Solutions Team, CBT Communication Engineering, Getxo, Spain*

Keywords: Global Software Development, Collaboration, Tools, Tool Integration, Challenges, Solutions.

Abstract: Global, collaborative and distributed development is increasingly common in software development. This way of working, widely adopted by big corporations has been also included lately in smaller organisations including SMEs. The traditional product and software development technologies do not correctly support this way of working, e.g., time and cultural differences add new requirements for these technologies. The Prisma project[1] aims to provide the tools, experiences and guidelines to help companies that face these issues in their everyday project management. One of its main outcomes is the PSW (Prisma Workbench). PSW is a platform and a set of tools that addresses the issues that need to be overcome in adoption of a collaborative and distributed development in organisations. This paper will present the results of the study performed during the Prisma Project and highlight the features of the PSW that will facilitate global collaborative software development teams to work more effectively in this kind of environments.

1 INTRODUCTION

In the perspective of growing size and complexity of embedded systems, price competition and demand to bring out new products to the market faster, many companies realise that they are not able to develop all the required functionality in one location only, or by themselves. As a result, companies establish new development units, e.g. to low cost countries, and suppliers specialize in specific functionality or specific skills (so called core competencies) which they can sell to others. This distributed approach has been adopted in the last years also by SMEs which want to profit from the same benefits. This is clearly visible in the growing numbers of the outsourcing constructions. Companies have to outsource large parts of their developments to specialized suppliers, often globally distributed. Related skills are no longer available in their own organizations and they need to manage a complex situation of many partners, sub-contractors, suppliers, software platforms beginning the etc.

The trend is that the amount of collaboration continues to increase assuming new forms. At the main outsourcing used to be to achieve cost savings (Forbath, Brooks and Dass, 2008). In contrast, the strategy of some leading firms lately is to disperse R&D throughout the world and to truly collaborate with global partners on product innovation (Chesbrough, 2003) (Bass, Herbsleb and Lescher, 2007) (Booz, 2006). However, industry suffers from a rapidly decreasing productivity as a consequence. Once people start working together, they face other problems. Such problems are e.g. integration of different software development processes or asynchronous collaboration because of time difference among partners. Global and distributed projects require new processes, tools and solutions to solve their particular problems.

Merlin and TWINS projects have done substantial research in the area of finding out solutions for collaborative product development.

Challenges encountered during collaborative development by companies were collected in the areas of Collaboration management, change management, requirements engineering, architectural design, integration and testing, configuration management, co-operative work etc. As a result, a web-based handbook was published and a Merlin Toolchain was developed. Merlin Handbook (Parviainen, Eskeli, Kynkäänniemi and Tihinen, 2008) was developed by collecting these

[1] http://www.prisma-itea.org/

Eskeli J. and Maurolagoitia J..
GLOBAL SOFTWARE DEVELOPMENT: CURRENT CHALLENGES AND SOLUTIONS.
DOI: 10.5220/0003444200290034
In *Proceedings of the 6th International Conference on Software and Database Technologies* (ICSOFT-2011), pages 29-34
ISBN: 978-989-8425-76-8
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

challenges, and it also has a large number of solutions that help in tracking them.

In Prisma project, we are focusing on more problem areas of Global software product development while also increasing the level of coverage for problems that were identified in Merlin project. In this paper an inventory of the most common problems and solutions present in literature are described.

2 PROBLEMS RELATED WITH GLOBAL SOFTWARE DEVELOPMENT

In this section we present some problems present in global software development. These were identified from literature, and from industrial partners experiences in Prisma project.

2.1 Problems Presented in Literature

Collaborative development cannot only be found in the case of offshoring. Nonetheless it can be considered as a borderline for this way of managing development projects. Here is the summary of offshoring challenges, which could be an interesting starting point (Carmel and Tija, 2005):

- Communication Breakdown: Human beings communicate best when they are interacting face-to-face. Communication over distance frequently leads to misinterpretation. In multi-site project, people face communication problems due to e.g. language barriers and unavailability of resources. In such cases they prefer to communicate over e-mails or over chat servers. In these communication patterns if something is misinterpreted or not clarified properly it can lead development to wrong direction.
- Coordination Breakdown: Software development is a complex task that requires many small and large adjustments. People need to co-ordinate well when they are working on a common task. It is easier to co-ordinate spontaneously in face-to-face conversation. In geographically distributed projects, these small adjustments do not take place or it is not easy to make these adjustments. Thus, problem solving gets delayed again and again, or the project goes down on the wrong track until it becomes very expensive to fix.
- Control Breakdown: Successful management control takes place when managers can roam around to see, observe, and dialogue with their staff. This type of management by walking around (MBWA) is not feasible when software developers are located many kilometres away. Managers have to rely on telephones, E-mail and other communication means (e.g. chat servers) and this can provide a less clear picture of the development status.
- Cohesion Barriers: Working groups are composed of dispersed individuals. Such groups are unlikely to form tight social bonds, which are a key to a project success.
- Culture Clash: Each culture has different communication norms. The result of these differences is that in any cross-cultural communication, the receiver is more likely to misinterpret messages or cues. Hence, the familiar complaint of miscommunication across cultures is always present.

In taking a step further towards a more technical point of view several references have been found to problems connected with the requirements gathering and management (Sinha, Sengupta and Chandra, 2006)(Damian and Zowghi, 2003). Requirements gathering and management is one of the most collaboration-intensive activities in software development. Geographic separation makes it much more difficult to hold effective discussions around requirements, to manage requirement changes across several sites, and to preserve and harness project knowledge.

Desouza, Awazu and Baloj (Desouza, Awazu and Baloj 2006) point out the highly knowledge intensive efforts required in global software initiatives and the management challenges it poses for organizations. The use of different tools and data formats, for instance, makes it difficult to easily exchange information and development artefacts (work products) (Gao, Itaru and Toyoshima, 2002).

Similar issues can be found in other business areas such as manufacturing were are looking to IT to provide the techniques and tools for constructing virtual organizations that will support electronic collaboration (Fecondo, 2006).

2.2 Problems Identified for Prisma Industrial Partners

One of the main activities in Prisma Project has been to identify the problems that have been found by all industrial partners while managing distributed collaborative software development projects.

- Learning Curve: People do not like to change the technology or tool on which they are developing a product. They are not familiar with new tools and technology and they tend to resist when they have to change their working platform. Thus learning curve is very low.
- Poor interoperability between tools: For example, when data is moved from requirements management tool to a test tool, defects are easily introduced. Different partners may use different tools and when the data is integrated it can result into many errors and defects.
- Responsibilities and roles are not properly defined: People do not know to whom they should report and who is responsible for what task. In such cases the problem is that the escalation mechanisms are not clearly defined.
- Lack of knowledge of standard solutions: Sometimes developers start creating the same solution which has already been implemented as a standard solution. This leads to total waste of resources. Before starting the development, a proper background check for the product is not provided. Developers don't know why are they creating some product and what is the purpose behind creating it.
- Resource management: It is very difficult to manage resources in multisite project environment. People with right skills and real competence are always busy with loads of work. Thus it is very difficult to start up work, when proper resources are not available or the information about them is unavailable..
- Cost of currently available tools: Nowadays the market has a number of tool providers that supply solutions for managing collaborative projects. Most of these solutions will only communicate with tools from the same provider limiting the organizations options. The cost of this investment is sometimes higher than companies can afford.

2.3 Current needs of Prisma Partners

The current development processes for the project partners have been reviewed along with the tools and methodologies that are being used. This review has allowed extracting the set of requirements and needs that should later be implemented by the tools and methodologies that are the result of the Prisma Project. A highlight of these requirements is the following:

- Requirement capture process: the Requirements are captured by the partners during meetings with their customers. Finally these requirements are collected in documents. The tools used in this process are: Trac[2], SQS AgileREQ[3], Focal Point[4], MS Sharepoint[5], MS Excel sheets and DOORS[6]. The needs for a requirement capture tool include having a common and unique requirement repository that implements traceability mechanisms with other information items (e.g. other requirements, bug reports, related test cases and requirement information sources).
- Requirements review process: the requirement review is an iterative process throughout the project, in meetings between the stakeholders. If the requirements have to be modified, these changes are documented. The tools used in this process are: Trac, custom made self developed tools (Formal Peer review process) and MS Sharepoint. The needs for a tool that helps during this vital activity are the possibility to keep track of a full history of review comments and access to metrics (e.g. review effort or defects found).
- Traceability process: the tools used by the partners only manage traceability between error tickets and test cases. The tools used in this process are: Subversion[7] and TortoiseSVN[8], DOORS, SVC, SQS TestWORKFLOW[9], and Excel sheets. Because traceability is a key task in collaborative software development, the main need of the partners was to have the possibility to assign and review the traceability among all the information items in the project (e.g. requirements, test cases and reports, bug reports and others).
- Testing process: the partners test models and prototypes, developed modules, the integration of the developed modules, and the final product (functional testing). In some cases the partners simulate the behaviour of the desired product by using emulators. The tool used in this process is mainly SQS Test WorkFlow. The needs for testing tools include a way to determine which tests are required to validate a

[2] http://trac.edgewall.org/
[3] http://sqs.es/en/solutions/agilereq/
[4] http://www-01.ibm.com/software/awdtools/focalpoint/
[5] http://sharepoint.microsoft.com/
[6] http://www-01.ibm.com/software/awdtools/doors/
[7] http://subversion.apache.org/
[8] http://tortoisesvn.tigris.org/
[9] http://sqs.es/en/solutions/testworkflow/

release, specification of various test configurations, being able to repeat standard set of tests quickly, and create automatically test reports, easy definition of test scripts, continue regression if an unexpected problem or defect is encountered, and automatic defect reporting with attached information of test case, error condition, and test data.

- Metric capture and analysis process: the metrics defined by the partners monitor e.g. test coverage and test results, efforts spent, number of errors revealed, requirement changes count, and time rates. The tools used in this process are: a custom self developed tool, Excel sheets and MS Sharepoint. Metric capture and analysis tools should improve the reliability of the releases, and also decrease the validation times, collect automatically metrics and generate reporting graphs and include overview and detailed views.

3 PRISMA WORKBENCH

Prisma Workbench (PSW) is one of the main results of the PRISMA project. PSW is an integrated environment that aims to support collaborative, Global Software Development (GSD). PSW builds on top of experiences achieved in Merlin and TWINS project (Eskeli and Parviainen, 2010), but has been completely redesigned. One of the major design goals has been to make PSW flexible and extendible by allowing for a configurable set of development tools tailored to individual partner or project needs. This is done so that the legacy tools already in use in the companies can be easily integrated to PSW. Continued use of familiar legacy tools is important for effective way of working that could be easily disturbed by a sudden change of tools. The initial set of tools integrated into PSW will originate from the Prisma project partners and open source.

PSW is an integration solution that does not aim to replace the existing tools, but rather to provide its services in parallel to the legacy tools. Due to this decision PSW does not support modifying of tools' data; it is still done in the tools themselves. What PSW can do however, is to provide various (near) real time views into data. The views have been designed to alleviate issues (communication breakdown, control breakdown, etc.) encountered in collaborative, distributed development settings. The views provide visibility into project's progress by gathering and formatting data from the tools, managing connections between the data, and concentrating this information into easy to read dashboards (see Figure 2).

The connections between data (i.e. work products such as requirements) can be managed in PSW via traceability relations. Traceability relations indicate dependency between work products. For example, a relation between a requirement and a test case can indicate that the given requirement is validated by the related test cases. PSW provides means to specify these relations and views to visualize them. The relations can also be used in reports, e.g. to show requirements test coverage, requirements test status, etc. This kind of feature is especially useful in maintaining control of the project when the work products that should be logically dependant are managed in separate tools (and/or sites).

The views in PSW enable inspection of data stored in tools', but also provide notifications of important changes happening in the project. PSW contains a detection mechanism for integrated tools that provides information of additions, deletions, and modifications in work products that are stored in the tools. The notification mechanism also provides information of actions done in the PSW environment, such as tracing of work products. In the future this mechanism can be extended so that when a user logs into PSW environment the user is provided information of important issues that have happened since the user was last online.

The initial set of tools for PSW has been selected based on the Prisma project partners' preferences, but also on our previous experiences of these tools. The tool set consists mainly of open source tools: Trac for bug tracking, Testlink[10] for test case management, Subversion for version control, CruiseControl[11] as build tool, and OpenMeetings[12] for communications. A commercial AgileReq[13] tool from SQS is used for requirements management. All of these tools are connected to PSW using the tools' own APIs.

4 IMPLEMENTATION

In design of PSW implementation main considerations have been that the software components could be relatively easily changed and that the system can operate in distributed fashion.

[10] http://www.teamst.org/

[11] http://cruisecontrol.sourceforge.net/

[12] http://code.google.com/p/openmeetings/

[13] http://sqs.es/en/solutions/agilereq/

PSW consists of two main components, the server and the client. The server component integrates tools and provides its services to the client(s). The client component is the visual interface in to the tool integration.

The server is built on top of Apache Tuscany[14], which is a framework for building Service Oriented Architecture (SOA) solutions. SOA based architecture was selected because it promotes loose-coupling between software components, which in turn provides us the freedom to add / remove / change the components as we best see fit. Another reason was because the integrated tools can be distributed and the SOA based approach provides us easy access to the tools. As was previously mentioned, the server component does the actual tool integration by interfacing with the various tools.

The first step in integrating a new tool in to PSW is identifying the information elements (i.e. requirements in a requirements management tool) to integrate from the tool. In this step the integrator is creating a (usually) partial structure of the data maintained in the tool, based on the needs of the integration.

The second step is to establish a connection from the tool into PSW for interfacing. This is usually done using the tool's API; however PSW does not care how the integration is implemented.

In the third step a Java class is created which inherits the PSW tool interface definition. As defined by the interface, the class must implement several basic functions:

- A function which lists types of work products supported by the tool.
- A function which returns the name of the tool.
- A function which returns the work products with relations to each other as defined in the tool (e.g. requirements hierarchy).

For representing work products in PSW a separate class has been defined that maintains source tool, type of work product, unique id (anything that can be used as unique id in the tool), attributes, and relation to other work products.

In the current implementation the classes that implement the tool integrations are included at compile time. In the future the implementation could be easily modified so that third parties could place their implementations in e.g. .JAR-archives. The archives could then be loaded on the server start-up or even during operation.

Preliminary integration experiences gathered show that a tool can be added rather quickly; most of the effort will be spent in studying the tool's integration mechanisms (e.g. API). Based on the feedback received we have produced a step by step guidance, and a thorough description of the integration mechanism. In this case the guidance was given verbally by one of the developers.

The server also provides single-sign-on type of access in the tools for the users. Single-sign-on in this context means that the user's accounts in the tools are tied to the user's PSW account. Furthermore, it implements the traceability service which can be queried for work product relations and for creating new ones. The traceability mechanism is implemented in essentially the same fashion as in our previous work (Eskeli and Parviainen, 2010); no data is replicated but unique identifiers are used to identify the work products in the tools. The relations are stored in a relational database, MySQL[15].

To improve the performance, the data from tools is temporarily cached using Memcached[16]. This is necessary because some of the tool specific queries can take a long time to complete (e.g. due to amount of data, tool location), which would result in a poor user experience. When the cache is updated at specific intervals, the changes in the work products are detected and stored. The changes in the work products can then be queried using the notification service.

The client component consists of several JSR 286[17] portlets. The portlets implement the before mentioned views. The portlets produce their content from the services provided by the server component. The portlets have been designed so that minimal or no changes need to be done to them if the integrated tools are changed. Most of the implementation is in Java, but the portlets also use JavaServerPages[18] (JSP), and JQuery for asynchronous updates. The portlets are currently hosted in open source Liferay[19] portal software. One benefit of the JSR 286 specification is that it is possible to change the portal software to another portal that supports the same specification with relative ease.

5 FUTURE WORK

In our future work we plan to implement things we already mentioned (integration guidance, improved notifications), make existing functionality more robust, but also work on things we think could

[14] http://tuscany.apache.org/

[15] http://www.mysql.com/

[16] http://memcached.org/

[17] http://www.jcp.org/en/jsr/detail?id=286

[18] http://java.sun.com/products/jsp/

[19] http://www.liferay.com/

benefit people working in collaborative, globally distributed software development. We plan on using communication software (OpenMeetings or some other) in PSW so that the communication portlet can be placed in any context (e.g. requirements review), and then anyone who enters that context is automatically part of the discussion. Discussions are stored and rationale for decisions made can be traced later asynchronously by anyone who needs this information, without resorting to e-mails and phone calls. We are also planning on improved reporting & metrics facilities in the PSW to help project managers can gain better control of the project.

To make integration of tools into PSW easier we are inspecting existence of standardized or de-facto tool interfaces that we could implement in PSW. In this case a tool implementing such an interface could be connected to PSW with minimum effort. We are also considering adding distributed version control software (e.g. GIT, Mercurial) to the initial tool set to complement centralized version control (Subversion).

6 CONCLUSIONS

In this paper the concept of Global Software Development was introduced and described as a growing organisation paradigm for software development companies. The main issues that this paradigm includes have been highlighted based on the research in literature and on the results of the Prisma project. The goals of Prisma are to provide the tools, experiences and guidelines to help companies that face these issues in their everyday collaborative project management.

Prisma Workbench PSW is one of the main results of this project and it has been presented as a flexible and adaptable platform the will allow companies to integrate their own existing tools in order to improve project management in a globally distributed organisation by raising awareness of project happenings and through improved co-ordination of activities. In Prisma project, the integrated set of tools was selected based on the partners' preferences. During the remainder of the project PSW will be extended as mentioned in the future work section. PSW will also be tried out in industrial setting where experiences of its usage will be gathered.

ACKNOWLEDGEMENTS

The authors would like to thank the partners involved in the ITEA2 Prisma project for their contribution and inspiration.

REFERENCES

Forbath, T., Brooks, P., Dass, A., 2008. Beyond Cost Reduction: Using Collaboration to Increase Innovation in Global Software Development Projects. *In IEEE International Conference on Global Software Engineering*.

Chesbrough, H., 2003. Open Innovation. Boston, MA., HBS Press

Bass, M., Herbsleb, J., Lescher, C., 2007. Collaboration in Global Software Development Projects at Siemens: An Experience Report, *In ICGSE '07 Proceedings of the International Conference on Global Software Engineering*.

Booz A. H., 2006. Globalization of Engineering Services", NASSCOM

Parviainen, P., Eskeli, J., Kynkäänniemi, T., Tihinen, M. 2008. Merlin Collaboration Handbook - Challenges and Solutions in Global Collaborative Product Development. *In Proceedings of ICSOFT (SE/MUSE/GSDCA)'2008*. pp.339~346

Carmel, E., Tija, P., 2005. Offshoring information technology. s.l. Cambridge University Press

Sinha, V., Sengupta, B., Chandra, S., 2006. Enabling collaboration Distributed Requirements Management, September/October 2006, *IEEE Software*, pp. 52-61.

Damian, D., Zowghi, D., 2003. Requirements Engineering Challenges in Multi-Site Software Development Organizations, *Requirements Engineering Journal*, Vol. 8, pp. 149-160.

Desouza, K. C., Awazu, Y. and Baloj, P., 2006. Managing Knowledge in Global Software Development Efforts: Issues and Practices, *IEEE Software*, pp. 30-37.

Gao, Jerry Z., Itaru, F., Toyoshima Y., 2002. Managing Problems for Global Software Production – Experience and Lessons., *In Information Technology and Management*, Vol. 3 (1-2), pp. 85-112.

Fecondo, G., et al., 2006. A Platform for Collaborative Engineering IEEE Software, pp. 25-32.

Eskeli, J., Parviainen, P., 2010. Supporting Hardware-Related Software Development with Integration of Development Tools. *In 2010 Fifth International Conference on Software Engineering Advances.*

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS