PSW: A Framework-based Tool Integration Solution for Global Collaborative Software Development

Juho Eskeli
VTT Technical Research Center of Finland
Oulu, Finland
Juho.eskeli@vtt.fi

Jon Maurolagoitia
CBT Communication Engineering
Getxo, Spain
jmaurolagoitia@cbt.es

Carmen Polcaro
Innovalia Association
Bilbao, Spain
cpolcaro@innovalia.org

***Abstract*—The market of solutions for collaborative and distributed software development offers currently a wide range of tools that support specific tasks involved in these kind of projects. Several solutions aim to support the whole development process in a single tool or via groups of tools by providing distributed teams the possibility to share and connect information and to use common interfaces. Nonetheless, every one of them includes some disadvantages that lessen their value for companies that use them across their distributed development projects. In this paper the authors will highlight relevant issues associated with collaborative and distributed software development projects. Prisma Workbench will be presented as the framework to overcome many of these issues and to provide a compelling option for teams to integrate their existing tools into a complete collaborative solution. Currently Prisma Workbench is being tested by the partners involved in the ITEA2 PRISMA Project and some of the first feedback will be presented as well.**

Keywords-collaborative software development; global software development; collaboration; tools; tool integration

I. INTRODUCTION

Collaborative and distributed software development is currently one of the most common ways of facing the development for many applications that due to its complexity or size require a large team working together [1]. The level of distribution for each group of the team can vary from different departments of the same company located in the same building to the case that several companies' located in completely distant regions of the world participate in a common development. The motivation to adopt this organizational paradigm can vary from case to case: cost reduction, collaboration between reference centres or using this as a way to increase the innovation inside the company [2]. The number of cases that can be found in the industry is enormous [3][4].

A distributed software organization model brings problems to the development process that have to be addressed with specific methodologies or tools. The most relevant that could be identified as part of the PRISMA Project[5], previously to the development of Prisma Workbench (PSW) [6], are highlighted here:

- Communication Breakdown: the barrier of not being able to discuss issues and agree on specific topics face to face leads to delays in the development process.
- Coordination Breakdown: can happen in a project where people don't know each other or don't have the possibility to interact continuously to adapt project planning. The chances of the project to go on wrong track are higher and following of planning is difficult.
- Control Breakdown: For project managers, having a clear view of the status of a project when the team is distributed in different locations and work in different time zones can be a really challenging task. The level of control that the project manager will have is not as deep as in a non-distributed scenario.
- Cost of currently available tools: currently a number of providers offer their commercial solution for collaborative development. The price of implementing these solutions in companies is sometimes an obstacle.
- Poor interoperability between tools: in a case where each team is using their own tools, integration between the tools is difficult and most of the times impossible. For this reason manual copying or exporting of data from one tool to another is often needed.
- Lack of traceability: during the development project information elements are created which traceability should be maintained throughout the whole process. These elements include e.g., client requests, system requirements, test information, bug reports, and so on. Having no connection between the tools that manage each of these elements makes the traceability maintenance an effort consuming task.

 ISBN: 978-1-61208-165-6

Nowadays, the market of tools that support specific tasks of the development process is very large. In most cases their learning curve is high. Therefore, teams feel reluctant to include a new tool or change the tools that they are currently using as part of their development process although this could sometimes lead to a better integration with the rest of a distributed team.

Another type of tools, which will be discussed in chapter V of this paper, presents a global solution that supports the whole development process. As mentioned before, these solutions include sometimes a price tag that not every company is able to pay, especially in those cases where SME's are involved.

Prisma Workbench, the solution proposed in this paper is a tool integration framework designed for collaborative distributed software development. This framework allows connecting of software development tools to create company specific software development environment instances. In this paper the solution is presented from instance point of view; how it can be used with a particular set of tools. The tool set mentioned consists of tools proposed by the PRISMA project partners.

PSW fills the gaps that exist in the current collaborative software development environments. It allows distributed teams to integrate their own existing tools and link data among them. PSW provides the visibility of how the project is running and what every group is doing to the whole development team as if everybody would be working in the same room.

II. RELATED WORK

Wasserman [7] defines tool integration as follows: 'tool integration is intended to produce complete environments that support the entire software development lifecycle.' In our vision tool integration can be used to provide a consistent software development environment using tools that were not planned to be used together initially. Furthermore, with the help of suitable tool set a notable part of software development lifecycle can be supported. Thus, the vision is not entirely separate of what Application Lifecycle Management (ALM) tools attempt to provide. According to Kääriäinen [8] ALM can be understood as coordination of activities and the management of artefacts such as requirements, test cases, etc. during the lifecycle of a software development project.

Schwaber [9] and Shaw [10] mention that the type of ALM solutions at that time could be divided into single vendor (e.g., IBM Jazz), multi-vendor (e.g., Eclipse, ALF), and single repository approaches. In single vendor approach a vendor has built a framework where other vendors can build integrations. In multi-vendor approach development and direction is driven by open source community (e.g., Eclipse, ALF). In single repository approach all the software lifecycle artefacts are managed in a single place.

According to the previous classification PSW is a multi-vendor platform. Furthermore, it is a framework integration based on tools' own repositories. As described by [11] framework-based integrations attempt to classify tools and provide integration between tool classes based on vendor-neutral interfaces and mechanisms. Furthermore, the framework-based approach aims to provide an integration environment and common look and feel without limiting the choice of tools [11].

As far as we know our solution is unique because it does not rely on any specific software development tool. Also, in theory the tool set could be extended to support notable parts of software development lifecycle using a suitable tool set. Modelbus [12] is a project of tool integration, but to our understanding the focus is mainly integration of modelling tools and study of model transformations. Also Eclipse Mylyn is advertised as ALM framework [13], but as far as we can tell it seems to focus largely on task management and integration of task / defect management tools.

III. FEATURES

PSW has been developed from ground up based on the experiences achieved from ITEA Merlin[24] and ITEA2 TWINS[25] projects. The previous Eclipse based tool integration has been described in detail in [14]. In case of PSW the main interface is via a web browser. This approach was chosen to decrease dependency on a particular technology/platform (Eclipse) and making it easier to use PSW in day-to-day operations (i.e. lower the barrier of deployment).

The solution proposed is a tool integration framework designed for collaborative distributed software development. In its current form it has been previously presented in [5]. PSW allows connecting of software development tools to create company specific software development environment instances. In this paper the solution is presented from instance point of view; how it can be used with a particular set of tools. The tool set mentioned consists of tools proposed by the PRISMA project partners.

PSW implements a repository neutral integration of tools. This means that the lifecycle data produced during software development process is maintained in separate tools. The benefit of this type of approach is that it has minimum impact on the company's current tool set. The caveat is that integrations to the tools have to be constructed on a per tool basis. However, there is no need to create point-to-point integrations between each of the tools because PSW acts as a hub where tools are connected via its integration interface.

For PSW one of the primary goals has been to make the integration of new tools as easy as possible. To get to this goal the following steps have been taken: designed integration mechanism for simple integration, provide example integrations, and created integration instructions. The integration mechanism has been described in [6]. The example integrations will be described later in this section.

The solution provides visibility of tools data in easy to understand dashboards (see Figure 1 and Figure 2) that can be customized based on the user's preferences. Furthermore, the framework handles user sign-in into the separate tools transparently. The solution also provides the means for the user to create links between different lifecycle items. These links can then be exploited in the reporting to e.g., demonstrate amount of defects in a build. The reporting

 ISBN: 978-1-61208-165-6

solution built into PSW allows users to customize their own reports.

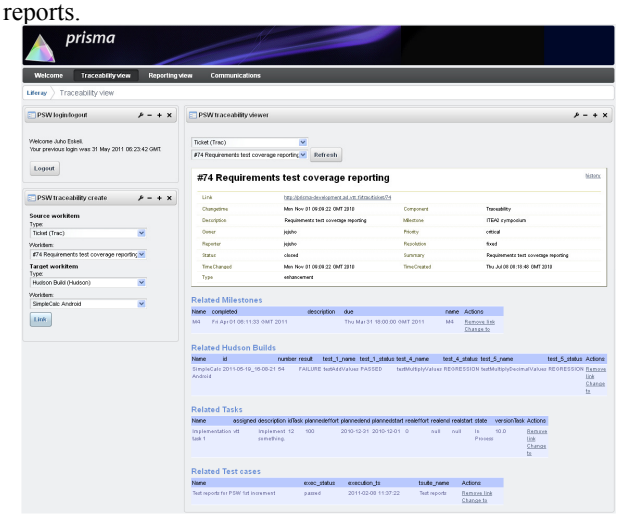

Figure 1. The traceability view showing a requirement and related work products

Figure 2. The reporting view showing a generated graph based on data retrieved from the integrated tools

To support collaborative, distributed development PSW provides means for asynchronous (chat) and synchronous (voice & video) communication with the help of a tool (OpenMeetings). The notifications system provides users up-to-date information about any important events (e.g., build status) in the project.

Although PSW can be connected to several other commercial tools or custom developed ones, the project team has made a selection of open source solutions that cover the complete development process. By using these solutions, companies will be able to start working together also if currently no tool is used for any of the requirements specification, development or testing tasks. The solutions that have been selected are the following:

- Edgewall Trac[15]: this tool originally developed for bug tracking has also been used a simple requirement management tool. As part of PSW it should be used for requirement management and bug tracking.
- Subversion: this versioning system is one of most popular in the open source community.
- Testlink[16]: This web based test management tool will support your test case and test data management.
- Openmeetings: with Openmeeting companies will be able to host their own audio and video conferencing solution.

IV. BENEFITS

PSW addresses many relevant issues related to collaborative and global software development. Some of these issues were extracted during the research done by the PRISMA Project and have been highlighted in the introduction of this paper. After taking into account the features available in PSW we propose how distributed development process could be dramatically enhanced using PSW:

- Communication enabler: the possibility to organize virtual meetings and link those to other information items such as requirements, test, etc. enables a centralized solution where every group of the team can refer to decisions made any time during the development process.
- Improved team coordination: by sharing the status of key information such as requirements or tests and providing an event log, every member of the team will be informed of what others are doing. This will help them to coordinate their own work according to the planning. In a scenario where groups work in different time zones this log will be sometimes the only reference to achieve this kind of coordination.
- Centralized project management: the dashboards provide information a project manager needs to have for a quick image of how the project is running comparing to the plan. It will also give access to more detailed view of specific tasks. Using only one tool (PSW) for overview will facilitate the continuous control of projects. The virtual meeting functionality will be a key tool for the interaction between project managers, group managers, developers and testers.

 ISBN: 978-1-61208-165-6

- Seamless integration between tools: PSW will enable tools from different vendors, located in distant location to integrate while maintaining their independence. As described in chapter VI, this integration can be done easily through standard REST or WS communication interfaces. The number of manual copying processes between tools to maintain the traceability throughout projects will be reduced and in most cases eliminated
- High level of traceability: One of the main benefits of PSW it the possibility to trace information from different tools as if all of it would be in one tool.
- Low cost of investment: By including PSW in your organisation, every group will still be able to use the same tools as they had done before since they will be integrated instead of being replaced. The investment needed is therefore much lower than in other cases where only tools from the same vendor can be linked.

The research performed as part of the PRISMA Project has included the analysis by the partners of the improvements achieved by using PSW in tasks that were supported before by independent tools or by no tools at all. Since the PRISMA project is still ongoing and will be finished by the end of 2011, only the preliminary results of this analysis can be presented here. Currently PSW is being tested in real distributed software development projects in order to extract the most valuable results. This analysis is being performed using the tools provided by default with PSW and described in chapter III. Some of these tools had already been in use for some time by the partners involved in the project.

The first comment that has been shared after starting this testing phase is that, although using the same tools as before, the information supported by those is not isolated anymore. The tool supported traceability helps every member of the team to have a clear view about how every information artefact is related with the rest.

The centralized reporting tool has been identified by project, development and test managers as one of the best features in order to review the status of the overall project. It is one of the main functionalities where PSW combines data coming from several tools and provides a higher level of information.

Future publications will detail the complete results from this analysis.

V. EXISTING SOLUTIONS AND APPROACHES

As mentioned before, the market offers currently a number of solutions focused on distributed and collaborative environment. As described below, most of them include any restriction due to being closely related with one development technology, provider or business model.

- Jazz: This solution from IBM is targeted to integrate the Rational line of tools which support several phases of the development process. These tools include Rational Requirements Composer, Build Forge and Quality Manager. Jazz also offers the Open Services for Lifecycle Collaboration (OSLC), an industry initiative to enable interoperability of tools developed by different vendors. Though promising, during the research performed in the PRISMA project, this interoperability was not achieved. Jazz is free to download from its site but currently it would be only useful for distributed teams that use Rational solutions.
- Teamforge: This webportal provided by Collabnet allows the collaboration of developers and IT project managers by proving the tools to plan and coordinate projects following agile methodologies. Collabnet features the management for user stories, source code integration, discussion forums, bug tracking and file and document sharing. Teamforge is licensed as a subscription based service. Although powerful, this solution forces every group of the team to use new tools and follow agile development methodologies which is not always the case in some companies.
- Application Lifecycle Framework (ALF): This Eclipse project proposal has been archived but its goal aimed to provide a logical definition of the overall interoperability business process. This technology handles the exchange of information, the business logic governing the sequencing of tools in support of the application lifecycle.
- Team Foundation Server (TFS): Microsoft offers this collaborative back end solution that can be connected with other Microsoft tools in order to exchange data among them. TFS does not have any user interface, rather it exposes web services which are the connection point between the tools. These include all the Visual Studio solutions but also Microsoft Project, Office or Sharepoint and cover almost the complete development lifecycle. As a disadvantage, teams where no Microsoft development tools are used will not be able to benefit from the TFS integration features.
- SourceForge.net: It claims to be the world's largest open source software development web site. They say that as of February, 2009, more than 230,000 software projects have been registered to use their services by more than 2 million registered users. SourceForge provides the following features for projects: discussion forums, wiki, version control system, file management and other tools more suited to open source projects.

VI. TECHNOLOGY BEHIND PSW

PSW consists of two main components: a server and collection of JSR 286 portlets. The server component integrates tools, implements some basic functions needed by tool integration such as user management, and provides its services to the portlets (or other possible clients). The portlets act as the user interface.

 ISBN: 978-1-61208-165-6

The server is built on top of Apache Tuscany[17], which is a framework for building Service Oriented Architecture (SOA) solutions. The framework takes care of runtime handling (initialization, termination, etc) of services. SOA was selected because it promotes loose-coupling between software components. Loose-coupling is useful because it provides us the freedom to add / remove / change the tools as needed. Yet another reason was because the SOA based approach provides us easy access to the distributed tools.

The integration mechanism of PSW has been described in [6]. A new tool can be integrated by creating a Java class that implements a Java interface definition provided by us. In the interface definition there are specific functions that need to be filled in; i.e. to get all work products (e.g., requirements) from the tool. What happens here is that the integrator creates a glue code that connects the data from the tool to PSW. The actual data from the tools can be fetched via any means supported by the tool, e.g., using REST or WS. Example integrations and guidance are provided to make the integration as easy as possible.

The server also takes care of authenticating the users to the tools. In essence a user's account for the tools is tied to the user's PSW account. Furthermore, it implements a traceability service which can be queried for work product relations and for creating new ones. The traceability mechanism is implemented so that no data is replicated. Instead unique identifiers are used to identify the work products in the tools, and the relations are stored in a relational database, MySQL[18]. The information artefacts are maintained in the original tool repositories.

For improved performance the data from tools has to be temporarily cached, for which Memcached[18] is used. Caching is needed because some of the tool specific queries can take a long time to complete (e.g., due to amount of data, tool location). The cache is updated at definable intervals. During an update the changes in the work products are detected and stored. The changes can then be queried using the notification service and shown in the user interface (i.e. portlets).

The user interface consists of several portlets implemented following the JSR 286[**Error! No se encuentra el origen de la referencia.**] standard. The views (e.g., traceability, reporting) are implemented via one or many portlets and use the services provided by the server to produce their output. The portlets have been designed so that minimal or no changes need to be done if the set of tools is changed. The techonologies used are Java, JavaServerPages[20] (JSP), and Javascript (JQuery etc.). For current implementation Liferay[21] portal has been chosen to run the portlets since it supports the JSR 286[22] standard. Nonetheless any other platform which support this standard could be used

The reporting feature is the most recent addition into PSW. It enables users to build their own customized reports. An existing implementation (BIRT[23]) was studied and found promising; however the effort needed to implement custom reports with it in portlets was considered to be too much compared with the result. The reporting feature enables users to filter the data (e.g., from which tools, what type of work products) they use for the reports. Some rudimentary manipulation of the data can also be performed e.g., addition or grouping of values. Existing traceability information can also be used to create e.g., requirements test coverage report. The plot types supported are currently bar, line, and pie chart. New types can be easily implemented with the library that is responsible for generating the charts. Additionally, the parameters used for creating the report can be stored for further usage, e.g., recurring reports. Reports with data can also be stored, named, and dated for reference. Finally, the reports can be exported in CSV and PDF formats.

VII. CONCLUSION

In this paper the authors have presented relevant issues that development teams face when a distributed organization model is adopted. These issues, which were identified as part of the research of the PRISMA Project, have been the motivation to develop PSW, a solution that allows the integration of a heterogeneous number of tools in order to collaborate and exchange data while maintaining their independence.

Solutions for collaborative software development that are currently available have been described, highlighting the advantages of PSW among them.

PSW features, technology background and benefits have been also thoroughly explained in order to make clear how using this solution in a distributed and collaborative environment could dramatically reduce the impact of this organization model in software development projects.

ACKNOWLEDGMENT

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

REFERENCES

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

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

[3] M. Bass, J.D. Herbsleb, C. Lescher, "Collaboration in Global Software Development Projects at Siemens: An Experience Report" , 2007 , IEEE, International Conference on Global Software Engineering.

[4] Booz Allen Hamilton, "Globalization of Engineering Services", August 2006 , NASSCOM

[5] Prisma Project website http://www.prisma-itea.org/

[6] J. Eskeli, J. Maurolagoitia, "Global Software Development: Current Challenges And Solutions.", 2011. ICSOFT

[7] A. Wasserman, "Tool Integration in Software Engineering Environments", Springer-Verlag, Berlin, International Workshop on Environments, pp. 137-149, 1990.

[8] J. Kääriäinen, "Towards an Application Lifecycle Management Framework", VTT Publications, Dissertation, 103p., 2011.

[9] C. Schwaber, "The Changing Face of Application Life-Cycle Management", Forrester Research Inc., August 2006.

 ISBN: 978-1-61208-165-6

[10] K. Shaw, “Application Lifecycle Management for the Enterprise”, Serena Software, White Paper, http://www.serena.com/docs/repository/company/serena_alm_2.0_for_t.pdf, April 2007. (available 24.5.2011)

[11] J. Pederson, “Creating a tool independent system engineering environment”, In: IEEE Aerospace Conference, 8 pp., March 2006.

[12] C. Hein, T. Ritter, and M. Wagner, “Model-driven tool integration with modelbus”, In Workshop Future Trends of Model-Driven Development, 2009.

[13] http://www.eclipse.org/mylyn/ (read 27.05.2011)

[14] J. Eskeli & P. Parviainen, “Supporting hardware-related software development with integration of development tools”, Proceedings - 5th International Conference on Software Engineering Advances, ICSEA 2010, IEEE Computer Society, pp. 353 – 358, 2010.

[15] Edgewall http://trac.edgewall.org/

[16] Teamst http://www.teamst.org/

[17] Apache Tuscany http://tuscany.apache.org/

[18] Mysql http://www.mysql.com/

[19] Memcached http://memcached.org/

[20] JSP http://java.sun.com/products/jsp/

[21] Liferay http://www.liferay.com/

[22] JSR286 http://www.jcp.org/en/jsr/detail?id=286

[23] BIRT http://www.eclipse.org/birt/phoenix/

[24] Merlin-project http://virtual.vtt.fi/virtual/proj1/projects/merlin/icgse.html

[25] TWINS-Project http://www.twins-itea.org/

 ISBN: 978-1-61208-165-6