Collaborative Software Development Process for Geographically Distributed Teams

Andrea Pinto[1], Ana Carina M. Almeida[1,2], Elisabeth Morais[1]

[1]Centro de Estudos e Sistemas Avançados do Recife (C.E.S.A.R)
Rua Bione, 220 - Bairro do Recife – CEP 50030-390
Recife - PE - Brasil

[2]Universidade Federal de Pernambuco (UFPE) – Centro de Informática
Recife – PE – Brasil

{andrea.pinto,acma,beth.morais}@cesar.org.br, acma2@cin.ufpe.br

Abstract. *Several companies have started to work with geographically distributed teams due to cost reduction and time-to-market. Some research indicates that this approach introduced new challenges, because the teams work in different time zones and have possible differences in culture and language. A standard way to develop a system is a critical factor for project success, because it ensures interaction and synchronism among sites, besides that increases team productivity. In this way, the main goal of this paper is to describe the solution adopted by a Brazilian team to develop code and write documentation in a multisite project environment.*

1. Introduction

Over the last decades, many companies have started to work developing software with geographically distributed teams. A lot of advantages encouraged them to implement software in multisite [Jorge, Rafael 2008]. The main factors that have driven distributed software development (DSD) are:

- Differences in the development cost among offshore centers – The market demand for system development is bigger than the number of engineers available [Damian, Zowghi 2002], due to the significantly increase of the engineer salary in specific areas as well as the software cost. Besides, some governments subsidize a taxes deduction for companies in order to stimulate Information Technology (IT) business in their countries reducing the software development cost [Jorge, Rafael 2008], [Carmel 1999].
- Time-to-market – Creating products with teams in different time zones in order to accelerate development time by using the follow-the-sun concept [Jorge, Rafael 2008].
- Qualified and available engineers to develop software [Jorge, Rafael 2008].
- Staying close to the customer in order to properly know their business and needs [Jorge, Rafael 2008].

On the other hand, the DSD approach has many challenges to make the project successful. Mainly for the project managers that need to synchronize the activities and communication among different sites with different time zones, cultural aspects,

language and sometimes different development process among sites [Jorge, Rafael 2008], [Herbsleb, Audris, Thomas, Rebecca 2000].

Ramasubbu's research [Ramasubbu, Balan 2007] reveals that, even in a high process maturity environment, a distributed team can have a low productivity and its effect can be minimized by structured software engineering process adapted for DSD projects.

The aim of this paper is to present the Contribution Processes, created for multi-site development in order to implement an IDE (Integrated Development Environment) software platform and elaborate a developer guide for supporting platform extension. We describe the process flow, tools, roles and responsibilities, besides that, the main challenges and lessons learned.

This article is organized on the following way: section 2 describes the project purpose, environment and main challenges; section 3 shows development contribution processes; section 4 presents the lessons learned and section 5 shows the work conclusion.

2. Project Purpose and Challenges

The application domain discussed in this article is an IDE software platform for different Mobile Platforms. We have 5 sites in Brazilian territories: 2 sites in the northeast area and 3 in the southeast. Each team has at least 10 developer engineers. The sponsor outsourced the project and he is located in the United States. The Figure 1 shows the sites localization.

Figure 1. Project's sites localization

Different technologies were used to implement the software, such as Java and C/C++. The software supports different operating systems, such as Windows, Linux, and Mac OS.

The Site 1 (Platform) was responsible for creating a framework on top of the Eclipse Platform in an extensible way, and for defining, together with other sites, a set of features and standards to be extended by all sites.

The framework aimed to create a unified line of tools that gives a seamless development experience to external developers.

The other sites were responsible for extending the Platform framework through Eclipse extension points to create IDEs for different mobile platforms.

After a year of development, the Brazilian Site 1 proposed a new approach regarding this application construction since the Platform code was being extended by all Brazilian sites.

The proposal was to start the development in a collaborative way allowing everyone to cooperate in a systematic way to improve the Platform code. To accomplish this approach the Site 1 defined a collaborative process with a main objective; it should not be a process that would impact the development process of the other sites. This objective was important to keep the teams productivity. Besides that, the sponsor requested to use as much as possible open source tools in order to minimizing the project cost.

All sites communicate with each other as can be seen in the team distribution presented in the Figure 2, and the processes defined for all sites considered the team roles spread in the sites. The process activities were defined considering all the roles and the relevance of their participation in the project, in addition an activity would be executed by one or more roles distributed in the sites.

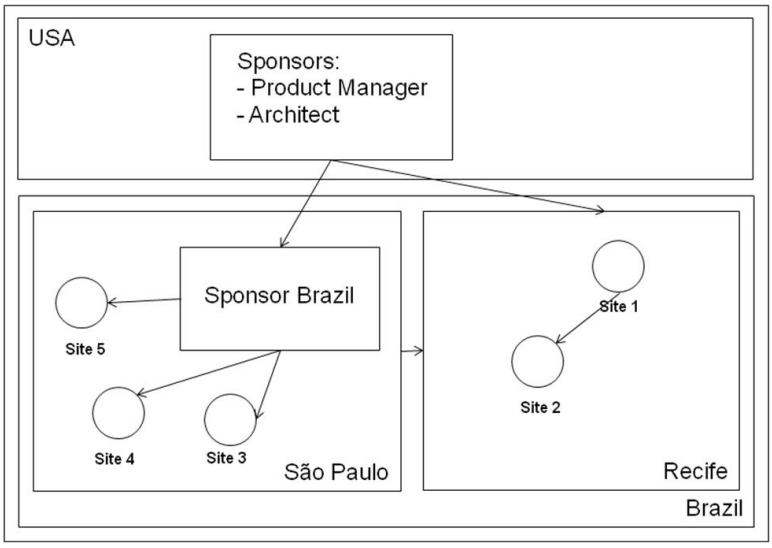


Figure 2. Project's team organization

This proposal was a step ahead and added more complexity to the project, besides the usual activities. The Platform project activities can be resumed as follow:

• Developing of the Platform project considering the needs of many sites developing applications that will be later integrated on it;

• Developing of applications that will be integrated in Platform in order to have a single IDE for a range of mobiles and technologies; these applications must follow Platform standards, and;
• Evaluating and integrating contributions from all sites in the Platform code.

Collaborative development software by auto-organized and distributed teams is not so trivial and requires definition of a development process considering a different context from traditional software development.

The processes presented in this article were defined considering the project specific needs mentioned above and a tracking mechanism was established to guarantee the process improvement and adherence. The processes were defined by the Brazilian sites and a kick-off meeting was performed involving all teams before to get the project started [Almeida, Junior, Carneiro 2009]. This is an important practice for providing alignment between all teams and preventing barriers that may block development of effective contributions.

The next section presents the processes related to the contribution activities from all sites to the Platform application's code and documentation.

3. Platform Contribution Processes

A well defined development process is a key point to any software project success; in projects whose main characteristic is the distributed development this affirmation is even more truthful regarding these projects complexities.

Models like CMMI [CMMI] includes in this scope specific practices to deal with distributed development, such as establishment of empowerment mechanisms, project's shared vision and integrated team structure.

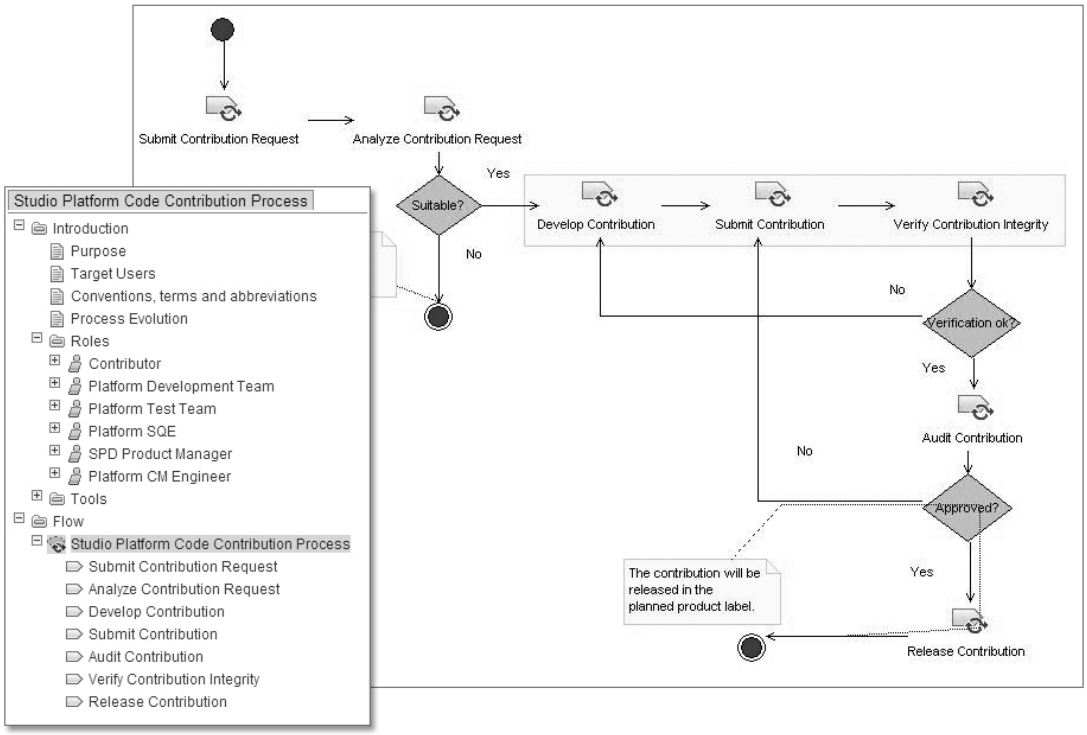
The Platform Contribution Processes, presented in the Figures 3 and 4, allows collaborative development and are used by Platform and other site teams. These processes are used to analyze the contribution requests in order to assure conformance with product purposes; to develop contributions according to specified requirements, standards and legal rules; verify contributions assuring product integrity; and finally release contribution according to the Platform features schedule and configuration management rules.

Figure 3. Code Contribution Process Flow

These processes have similar structure and common roles, they are:

• ***Contributor***: a person that collaborates with Platform, involved in new development or improvement. They are not part of Platform development team. They are originally part of development teams working for IDEs development, however they also aim Platform evolution for attending their necessities.

• ***Platform Development Team***: this team is responsible for developing and improving Platform.



• ***Product Manager***: responsible for analyzing contribution together with platform development team.

• ***Platform Test Team***: this team is responsible for testing Platform development and improvements performed by contributors.

• ***Software Quality engineer***: responsible for verifying if the contribution development followed source code contribution process.

• ***Configuration Management Engineer***: responsible for establishing baselines and release contribution.

Besides that, common tools were used, they are:

• ***IBM ClearCase Tool***: a proprietary version control tool, used to create and maintain product repository.

• ***Mantis Tool***: an open source change management tool through which the contribution requests are submitted and tracked during their whole lifecycle.

• ***Client Source Forge***: Open source version control tool, used to create and maintain product repository. It is generally adopted by open source projects.

• ***Black Duck Tool***: a proprietary code detection system used for avoiding plagiarism.

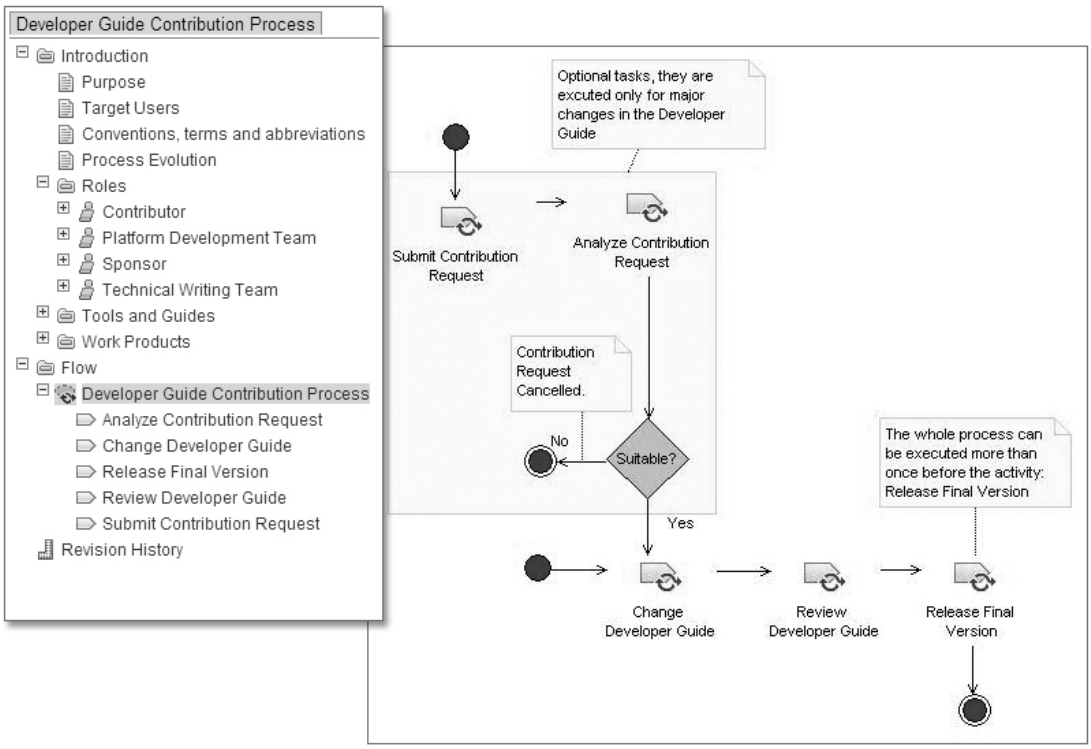


Figure 4. Developer Guide Contribution Process Flow

Both processes consider that a contribution is submitted through a request that is evaluated by a board responsible for checking conformance with platform purposes. This board is composed by product manager and platform development team. So, if a contribution request is suitable, it follows for development when the contribution will be properly integrated in the next software release. For a better understanding, the activities of the code contribution process presented in Figure 3 are described in detail below:

- ***Submit Contribution Request***: Contributors submit a contribution request in a change management tool, specifying what is going to be developed for Platform application. Contribution request must be unambiguous and well written, providing a good understanding.
- ***Analyze Contribution Request***: Platform Team and Product Managers analyze the contribution request verifying if it belongs to Platform framework purposes and register evaluation results. After the evaluation, the decision taken is included in the "Contribution Request". If the contribution request is suitable the Platform Team firstly verify the priority of the proposed contribution development and then assess the impacts of the contribution development regarding other Platform work products, such as requirements, design, tests, Open Source Software, Licensed Components and Developer Guide. If the proposed contribution does not belong to the Platform scope, the "Contribution Request" is cancelled and a justification is registered.
- ***Develop Contribution***: Contributors develop contribution assuring conformance with requirements, standards and legal rules, following the software development process adopted by their organization

•***Submit Contribution***: The Contributor submit the contribution in the change management tool, considering all impacted artifacts, specifying the work products locations for Platform team accessing them, being required to inform also the code scan report location.
•***Verify Contribution Integrity:*** Platform development team verifies the contribution integrity in the Platform, executing tests.
•***Audit Contribution***: After a successful verification, the Platform Team software quality engineer (SQE) audits the contribution before baseline establishment and evaluates findings. The SQE verifies if all the process activities were correctly followed, verifying inspections evidences for impacted software artifacts (documentation and code), code scan report, test design and results. In case of any gaps, they are described in the contribution. In this case, SQE decides with platform development team if the "Baseline Request" may be forwarded for the configuration management engineer for integration, considering that they can be resolved in subsequent contributions. The analysis and decision are documented in the contribution request. In case of no gaps or low risks gaps the audit is considered approved and SQE forwards "Baseline Request" for the configuration management engineer for integration.
•***Release Contribution***: After successful Platform SQE auditing, the baseline can be established. However, the Platform configuration management engineer consults the Platform development team to validate the label which the contribution will be integrated. If none was defined, the baseline request status is changed to "on hold", until a label have been defined. The integration and system tests are performed following the Platform development process. The contribution is released in the planned platform label.

Although collaborative processes should be defined in a light way however keeping the necessary formality for distributed development, some controls were inserted in these processes aiming quality. According to [Hecker 2000], properly organized and coordinated, distributed development can produce products faster and with higher quality than would be possible in an isolated effort. This can not only increase product functionality and quality but can also increase the value of the product as a platform for third-party developers and channel partners. Based on that, each site develops a contribution following their specific process; however their processes shall follow common inspection and quality politics specified by the client sites.

Both processes were specified using EPF – Eclipse Process Framework that allows reuse of the process elements. They were published in the Platform web site accessed by all teams through Internet.

4. Lessons Learned

This section describes the lessons learned about the distributed project focusing on the defined processes used by all sites.

It was noticed that the creation of both, code and developer guide contribution processes collaborated to improve the spirit of a single team, no matter where these were, supported by common tools.

Both processes were presented for whole team by a kick-off meeting. During these meeting, open ended questions were asked, in order to check if the attendees

understood the context, and also summarized the conversation in a meeting minutes sharing it with all.

Another important approach adopted was to allow the sites followed their own internal development process. In other words, they follow the collaborative process, when it is necessary to submit and evaluate if a component could be integrated in the Platform code. The intention using this approach was to keep teams productivity.

Regarding communication among development teams, Almeida [Almeida, Junior, Carneiro 2009] describes an interesting strategy to keep all sites in the same page. The Platform Project Portal was created to improve communication and share information, like, teams' features schedule and their release dates improving the visibility of the other teams, creating a unified product vision that can help in their features negotiation with the sponsors.

The processes were created using EPF which generate the output as HTML format allowing to share the process in an easy way with all sites through the project portal.

The technical documentation is stored on wiki site allowing the developers writing and sharing information about the framework architecture with others. It is a very interesting approach, because all sites were responsible to improve the technical documentation, contributing with the topic that they have more knowledge or feel more self-confident to document.

Last but not least, a very helpful good practice was the use of ambassadors (people who travel between sites). This approach helped a lot to establish trust and cohesion between the teams and sponsors [Almeida, Junior, Carneiro 2009].

5. Conclusion

The aim of this paper was to present the Contribution Processes, created for multi-site development in order to improve an IDE (Integrated Development Environment) software platform and elaborate a developer guide for supporting platform usage.

The collaborative processes defined, considering distributed teams working to improve the Platform framework, have contributed substantially to reduce the sites effort to working cooperatively, since their responsibilities and activities are clear defined and presented, as well tracked and improved by Site 1.

The processes did not add a bigger complexity to the usual sites development activities, an objective that was achieved by not changing the sites internal activities; the collaborative processes are mainly concerned to the activities of submitting, evaluating and accepting or not the sites contributions considering the standards that have to be followed and the project goals and priorities.

On the other hand it is important to understand the challenges that this approach introduces due to sites localization, cultural differences, distinct cultures related to processes usage, conflict of interests and the idiom.

References

Jorge, A., Rafael, P., Desenvolvimento Distribuído de Software (2008). Rio de Janeiro, Brasil. 2008, Elsevier.

Damian, D. and Zowghi, D. The impact of stakeholders? Geographical distribution on managing requirements in a multi-site organization (2002). In RE, pages 319–330.

Carmel, E. Global Software Teams – Collaborating Across Borders and Time-Zones. Prentice Hall, USA (1999).

Herbsleb, J., M. Audris, A.F. Thomas, E.G. Rebecca, Distance, dependencies, and delay in a global collaboration (2000), ACM, Computer Supported Cooperative Work, Philadelphia.

Ramasubbu, N., Balan, R. K. Globally Distributed Software Development Project Performance: An Empirical Analysis (2007). Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE), Dubrovnik, Croatia, September.

CMMI. Available at http://www.sei.cmu.edu/cmmi/.

Almeida, A., Junior, I., Carneiro, P., Managing Communication among Geographically Distributed Teams: a Brazilian Case (2009). SEAFOOD.

Hecker, F., "Setting Up Shop: The Business of Open-Source Software", IEEE Software (2000) available at: http://hecker.org/writings/setting-up-shop, Last access May/2009.