An Approach for Collaborative and Distributed Software Process Improvement (SPI)

Viviane Malheiros[1,3], Carolyn Seaman [2], José Carlos Maldonado[1]

[1]Instituto de Ciências Matemáticas e de Computação – USP
Caixa Postal 668 – 13560-970 – São Carlos – SP – Brazil

[2] Department of Information Systems – University of Maryland, Baltimore County
1000 Hilltop Circle, Baltimore, MD, USA, 21250

[3]Serpro – Serviço Federal de Processamento de Dados
Av. Luiz Vianna Filho, 2355, Paralela, Salvador/BA, Brazil 49010-000

Abstract. *Software Process Improvement (SPI) is an important challenge to organizations. Scenarios of geographically distributed software development highly reinforce key success factors to SPI. This paper presents an approach to support geographically distributed SPI initiatives. ColabSPI is a distributed and collaborative strategy and infrastructure to support SPI teams and developers in handling different phases of a typical SPI lifecycle. A prototype is presented together with some preliminary results and ongoing efforts.*

1. Introduction

For most organizations, software processes must be technologically competitive, adaptable and timely, and they must produce products that consistently meet customer and business needs (Florac et al., 1997). Good software processes should help output better software more cheaply and faster. Within such a scenario, Software Process Improvement (SPI) becomes an important challenge to organizations.

Different advances have been made in the deployment of SPI standards and models, for instance: CMMI (SEI, 2002), SPICE (ISO/IEC 15504, 2003), IDEAL (McFeeley, 1996), MPSBr (Softex, 2005) and The Experience Factory (Basili et al. 1994). However, the current problem with SPI is not a lack of a standard or model, but rather a lack of an effective strategy to successfully implement these standards and models (Niazi et al., 2005). Much attention has been paid to *which SPI activities to implement* instead of *how to implement* these activities efficiently.

Understanding how to implement SPI successfully is a hard task. From SPI literature and field observations, we've been identifying possible factors contributing to diminished SPI process performance and compliance regarding quality and time. So far, we've found and we are going to present elsewhere, that many influences on the success of SPI programs are related to coordination, communication and collaboration, and mostly to the degree of developers' motivation and participation in SPI initiatives. Scenarios of geographically distributed software development (distributed software teams) highly reinforce the need for dealing with such influences as:

- Processes for distributed software development (DSD) are more complex and challenging, as they are supposed to deal with communication and coordination issues. The effect of dispersion can be significantly mitigated through the use of structured software engineering processes (Ramasubbu and Balan, 2007) turning the development process a critical success factor for DSD (Prikladnicki et al., 2004) and making SPI extremely important on DSD context. However, continuous improvement (SPI) of complex processes are more complicated;

- As *developers' participation on SPI initiatives* is a key success factor, it is important to provide ways to geographically distributed developers contribute to process improvement.

Therefore, as DSD becomes more common the relevance of a distributed and collaborative SPI increases. Bearing this in mind, we propose a collaborative and distributed SPI approach to: (a) enhancing the communication and collaboration among SPI stakeholders; (b) increasing developers' participation in improving software development process; and (c) allowing coordination of SPI initiatives. The goal is to provide a strategy and a web-based project workspace to support: (i) SPI teams in handling process improvement proposals, (ii) process evolution; (iii) doubt clarifications and experience exchange; and (iv) management of SPI programs.

Our hypothesis is that **SPI programs** can benefit from a distributed and collaborative strategy and an infrastructure that not only creates a knowledgebase about a software development process and its improvements, but also allows SPI stakeholders to communicate and organize their work. Our focus is on large organizations that deal with DSD and aim to apply a standard processes to distributed development units.

By providing structured support, our approach may foster the emergence and progress of a cooperative environment for SPI. It can address major influences to SPI success or failure such as *knowledge exchange and support*; *staff involvement and motivation*; and *communication and collaboration*. To support our approach, we've considered that "software processes are software too" (Osterweil, 1997) and that more and more software companies make their process models available in intranets in order for them to be useful (Moe and Dyba, 2006). Important influences to our proposal are: concepts of (1) DSD, as in the Open Source development paradigm, and (2) Knowledge Management (KM) practices; and software infrastructure tools such as (3) Wikipedia; and (4) Bug Tracking tools.

This paper presents ColabSPI, an approach to promote geographically distributed SPI initiatives, supporting communication and collaboration, handling of process improvement proposals, process support as well as process documentation. It contributes to two major aspects of SPI: process evolution and compliance. The following sections bring an overview of our approach (Section 4), its requirements (Section 2) and influences (Section 3); and the ColabSPI in practices highlighting preliminary results and ongoing efforts (Section 5). Related works are discussed in Section 6. Finally our conclusions are presented in Section 7.

2. Key requirements for an SPI infrastructure

In previous work, an organizational structure that would improve developer' participation in SPI efforts was suggested (Malheiros et al. 2008a) based on experimental experiences, a first step towards distributed SPI. Further exploring SPI program issues we propose an infrastructure that would support a distributed and collaborative SPI work.

To define infrastructure requirements, we took key factors as our starting point. We have been collecting factors from primary and secondary studies, comparing them with our own observations in the field, and grouping them according to their nature and relationship into five groups: (i) Collaboration and Communication; (ii) Organizational Aspects; (iii) Compliance issues; (iv) Continuous Improvement Issues and (v) Staff Motivation and Participation. This grouping has given us a clearer idea of how to convert some of these identified factors into positive influences on SPI. Thishas inspired the definition of the infrastructure requirements. Recurrent factors were: (i) the need of staff motivation and involvement; (ii) the benefits of feedback, support for discussions and clear establishment of goals; and (iii) the availability of resources. Major requirements were identified (see Table 1).

Table 1 – Major requirements for the ColabSPI approach

1	Communication mechanisms that can enable cooperation such as discussion forums and mailing lists. In addition, the possibility of communicating events, news or SPI needs; publishing information on a message board or informing a particular community of interest;
2	Access to SPI information through a unique starting point;
3	Process under version control and the possibility of changing the process by more than one person from more than one place;
4	Collaborative SPI Strategy, with focus on empowerment and guidelines on how to contribute. Collaborative mechanisms may enhance the availability of resources;
5	Process Improvement Proposals (PIP) handling process (workflow, roles and functionalities) and the possibility of tracking each proposal status until its conclusion;
6	Collaborative support request handling;
7	Transparent backlog of PIP and support request allowing anyone to contribute to improvement analysis and to clarify questions; and
8	User spaces filtering information regarding user actions, such as a list of PIPs submitted by the user.

3. Major influences on our infrastructure

We have observed how different software development paradigms and approaches deal with issues like cooperation among distributed team members. We've focused our research on techniques from different domains that extensively rely on communication, collaboration and/ or coordination techniques. Considering that software process can be seen as software too (Osterweil, 1997), we've searched for inspiration from:

Distributed software development - We've transposed major characteristics of DSD to the SPI context, exploring how they could be extended to SPI. We were particularly interested in initiatives to building networks of software communities in large corporations (see Section 6). We have also considered reported issues related to global DSD in wider context: Carmel et al. (2001), Maindantchick and Rocha (2002); Prikladnicki et al., (2008). Some issues they arise may apply to collaborative and distributed SPI. We've also considered Open Source development characteristics (Reis and Fortes, 2002), as this is one example of distributed development. The following characteristics, already adapted to deal with process improvement instead of software development, are to be preserved: (i) the PIP management is distributed by the Internet; (ii) the process improvement is collaborative and decentralized; (iii) participation in SPI is motivated by personal interest of the user, or each one contributes according to his/her interest, yet anyone can contribute in some way.

Wiki contributions - A wiki is a collaborative website where its content can be edited by anyone who has access to it, provided that they have the required access levels. Wiki technology is relatively new and people are still experimenting with different ways of using it (Fogel, 2006).

KM practices – Developing software can benefit from many KM practices, and indeed several aspects of KM employed in software development have been studied. There are many tools to support some KM practices (e. g. contribution, knowledge acquisition, knowledge dissemination, collaboration) that can be useful to SPI. We are particularly interested in how to promote collaboration and improve participation benefiting from different skills. Knowledgeable people should be reachable for knowledge exchange, mentoring, advice or consultation. Building networks and "knowledge communities" powered by accumulated knowledge can be a good strategy to facilitate SPI. A systematic review (Bjornson, 2007) pointed that there are many open points regarding KM and software engineering. Aspects such as collaborative approaches, economic aspects, and knowledge mapping are not yet extensively explored in software engineering.

Bug tracking tools - As in software maintenance, it is possible to identify and deal with the weaknesses of a process version, converting them into improvements to the next version. In this way, one can relate error-handling management to PIPs. Both of them follow a workflow including submission (proposal), evaluation, approval (or rejection), implementation and deployment. In the software testing context this error handling is being supported by Bug-tracking tools. These tools could be customized to handle PIPs. According to Fogel (2006), the importance of a bug tracking system lies not only in its usefulness to developers, but in what it signifies for project observers. For many, an accessible bug database is one of the strongest signs that a project should be taken seriously. In that sense, an active PIP handling environment can indicate the wealthy of the SPI program.

4. ColabSPI: A collaborative /distributed SPI approach

Bringing all these influences together led us to explore collaborative development environments (CDE) as a good starting point for a distributed and collaborative SPI approach. A CDE is a virtual space wherein all the stakeholders of a project – even if distributed by time or distance – may negotiate, brainstorm, discuss, share knowledge, and generally work together to carry out some task, most often to create an executable deliverable and its supporting artifacts (Booch and Brown, 2002). It provides an integrated access for different mechanisms and tools, creating a virtual project space focused on the particular goal of a team. If we consider SPI as this goal we can imagine a virtual project space for the software development process, wherein Software Engineering Process Group (SEPG) members and developers can work together to carry out SPI. Likewise, they can negotiate, brainstorm, discuss and share knowledge about improvements toward a better software development process. Here we focus on software developers, SEPGs and specialist groups in their tasks of proposing, analyzing, and discussing process improvements; and implementing and deploying them, where they are physically separated and make use of the Internet as the medium for their interactions.

ColabSPI supports major activities of the SPI lifecycle. Guidelines for deploying SPI programs (e.g.: PDCA, IDEAL) usually suggests an iterative SPI, based on a gradual and evolving strategy. They also refer to identifying, developing and evaluating improvement opportunities to next cycles of the process. ColabSPI allows identifying and evaluating PIPs, developing such PIPs, according to their priorities, creating and deploying new versions of the process. It also allows SPI management (planning, monitoring and controlling). Handling SPI as a project is a common recommendation in models and guidelines. ColabSPI implements this recommendation.

ColabSPI contributes to most of the CMMI goals from *Organizational Process Focus* and *Organization Process Definition* process areas (see Table 2).

Table 2: Practices from CMMI supported by ColabSPI

Organizational Process Focus	Organization Process Definition
SG1 Determine Process Improvement Opportunities SG2 Plan and Implement Process-Improvement Activities GG2 Institutionalize a Managed Process GG 3 Institutionalize a Defined Process	SG 1 Establish Organizational Process Assets, particularly: SP 1.1-1 Establish Standard Processes and SP 1.5-1 Establish the Organization's Process Asset Library GG2 Institutionalize a Managed Process GG 3 Institutionalize a Defined Process GG 5 Institutionalize an Optimizing Process

ColabSPI allows all stakeholders involvement. Typical roles for distributed and collaborative SPI (Figure 1) were defined based on experiences in SPI at a large organization where SEPG members, engineering specialist groups and developers are distributed (Malheiros et al., 2008) and on Open Source development typical roles (e.g. *Specialist groups* hold rights equivalent to *committers'* rights). As a developer contributes to SPI he/she may be promoted from a general developer up to SEPG member, being responsible for core decisions on SPI (meritocracy).

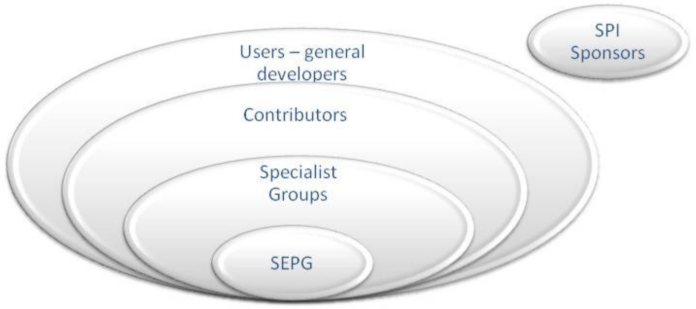


Figure 1: Typical roles for distributed and collaborative SPI

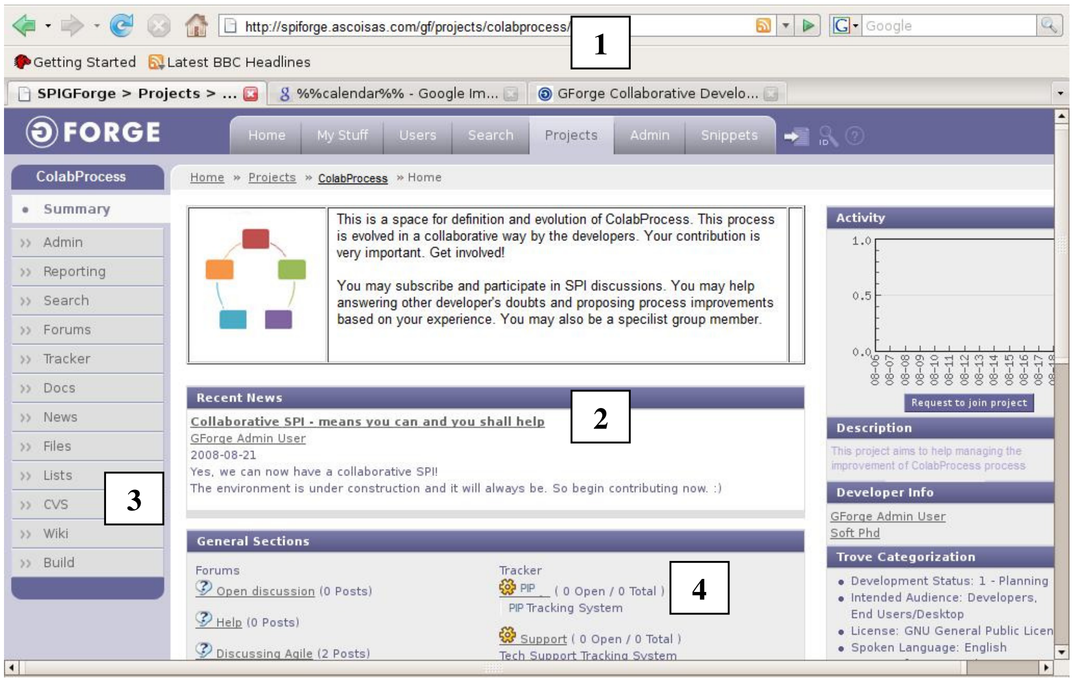
Mechanisms provided by our approach are classified into four major groups: (i) Collaboration and communication; (ii) PIP handling; (iii) Technical support handling; and (iv) Process documentation and maintenance;

Figure 2 shows a mockup of the SPI collaborative and distributed infrastructure. The screenshot highlights the virtual space of the ColabProcess "project". ColabProcess is an SPI project and all information about it can be accessed through one unique URL (see number 1, Figure 2). Once in the virtual project space, all communication and collaboration mechanisms are available to maintain and evolve the software development process: forums; news; trackers (bug-tracker like tools); reports; files from the software development process description itself; and historical data about it. Also, information about all contributors is available.

All functionalities to address the requirements (e.g. mailing list, forums, trackers, news) can be accessed through this unique starting point. Accessibility through the

Internet can enhance flexibility, distribution and easy connection of new tools.

At the main page, it is possible to know the latest news (see number 2, Figure 2) about the software development process. For instance, a SEPG can use this feature to inform when the next conference about SPI will take place or when a new major release of the process will be published. The software development process is under configuration management control (like software would be) and it can be accessed through a CVS link (see number 3, Figure 2). CVS keeps track of changes in process files and allows several stakeholders (developers, testers, SEPG members, etc.) to collaborate. All those stakeholders can be widely separated in space and/or time. Also, from the main page, it is possible to access all trackers related to the improvement of the software development process. For instance, it is possible to access all previous PIPs, their status and who is handling each PIP (Figure 2, id. 4).



This is a space for definition and evolution of ColabProcess. This process is evolved in a collaborative way by the developers. Your contribution is very important. Get involved!

You may subscribe and participate in SPI discussions. You may help answering other developer's doubts and proposing process improvements based on your experience. You may also be a specilist group member.

Recent News

Collaborative SPI - means you can and you shall help
GForge Admin User
2008-08-21
Yes, we can now have a collaborative SPI!
The environment is under construction and it will always be. So begin contributing now. :)

General Sections

Forums
Open discussion (0 Posts)
Help (0 Posts)
Discussing Agile (2 Posts)

Tracker
PIP (0 Open / 0 Total)
PIP Tracking System
Support (0 Open / 0 Total)
Tech Support Tracking System

Description
This project aims to help managing the improvement of ColabProcess process

Developer Info
GForge Admin User
Soft Phd

Trove Categorization
- Development Status: 1 - Planning
- Intended Audience: Developers, End Users/Desktop
- License: GNU General Public Licen
- Spoken Language: English

Figure 2 – The virtual project space of a software development process

The collaboration and communication module should augment SPI initiatives, by fostering the emergence and progress of a cooperative environment for SPI. According to Booch and Brown (2002), there is a spectrum of collaborative mechanisms that may be applied to a Web community, each with its own value. For our purpose, providing an infrastructure for SPI, we refer to some of these mechanisms particularly and add some mechanisms to Booch and Brown's list. Thus, "Mailing lists" will be applied for small groups with a common purpose, conversations that wax and wane over time, communities that are just getting started, and newsletters and announcements. "Message boards" will be useful for asking and answering questions, encouraging in-depth conversations, and providing context, history, and a sense of place. "News" will be useful for spreading novelties and communicating events. For instance, announcing the publishing of new software process releases or announcing that process version validations are taking place.

The PIP handling module is designed to help SEPG to keep track of reported

PIPs. It may be regarded as a sort of issue tracking system and was inspired by both Bugtracking tools, such as Mantis, and Serpro's tool for PIP handling, GM-PSDS (Malheiros et al. 2008). This module will tackle all functionalities related to posting, diagnosing, developing and concluding a PIP. Though majorly handled by one PIP tracker and its implemented workflow, we also intend to allow improvements through Wiki page editing tools, on a limited basis. Specialist groups could discuss assets and evolve their content through the Wiki. Also, specific components of the process, such as a guideline, could be evolved directly through a Wiki interface. However, placing the whole software development process into a Wiki could expose the process to the weaknesses of Wikis, such as: lack of a navigational principle, duplication of information, inconsistent target audience (Fogel, 2005) and lack of meta-data.

Regarding process support, any developer may answer requests and workflow states of support requests are limited to "submitted", "need information" and "answered".

Through ColabSPI, it is possible to represent (write) a software development process in many ways. In a previous work, part of our group developed Atabaque (Malheiros et al. 2008b), a free software tool that can be useful for organizations using web-based processes (available at http://sourceforge.net/projects/atabaque/). This or another tool can be applied to evolve the software development process. We only assume the process will be under configuration management control, whatever format adopted. The infrastructure must allow access to the configuration management system. Apart from that, process documentation is not the focus of this work.

5. ColabSPI in practice

Currently the SPI infrastructure is being forged based on an initial prototype. Our approach is independent from technology and it is focused on mechanisms rather than on specific tools. Even so, in order to experiment with our ideas in practice, and evaluate their real value, we've decided to customize one of the available CDEs for open source code. We've looked for previous tools that could completely or partially fulfill our high level requirements. An exploratory Internet search was conducted to find available collaborative tools that could be applied (customized) to SPI. Also, we considered Rehem (2008), where different options of CDE were systematically analyzed to be applied in the distributed software development in a large organization.

On such basis, the GForge CDE (http://gforge.org/) was selected as it's more suitable to the SPI approach requirements and because it is open source software, allowing complete manipulation of the several communication and collaboration mechanisms. In addition, being open source may facilitate the usage of the SPI collaborative and distributed infrastructure in other organizations and, particularly in the Qualipso competence centers (www.qualipso.org).

For analyzing ColabSPI we have created a SPI project in a GForge tool instantiated at a large software development software organization. In such instantiation, ColabSPI contains one general mailing list for SEPG, one mailing list for all developers, and one mailing list for each software engineering discipline covered in the software development process (e.g.: colabprocess-SEPG@spiforge.com). All mailing list discussions are recorded and available for further search in the process project environment. SEPG coordinator is the environment administrator. A forum was created to clarify doubts, with different entrances per discipline. A specific forum was created to "New ideas for the process". A question was posted in the message board related to process compliance: "Does your project recalculate software size when being closed?"

Focusing on collaboration the main PIP handling workflow presented in (Malheiros et al., 2008) was evolved to: allow every developer to see every PIP, even those under evaluation and registering contributions to a PIP at any stage. A new tool to support the new workflow is under construction with Mantis. The definition of a PIP handling customization with Mantis was first developed by Malheiros et al. 2007.

Atabaque and EPF Composer (Haumer, 2007) were tested to process documentation because they are open source tools; multi platform; generate the process in a web-based format and can be used integrated to a configuration management system for software process change control. Both presented satisfactory results on generating new versions of the process. The SPEM is suggested to modeling the process.

The prototype of the infrastructure was analyzed by some members of the SEPG group and was approved to be used in a pilot with all specialist group members. Piloting with specialist groups before making the solution available to all developers is compatible with SPI guidelines orientations.

In parallel, evaluation criteria are being detailed according to the GQM strategy (Basili, 1992) to measure quantitative the approach's value. Main factors that influenced our approach (briefly mentioned in Section 2) are the basis for defining the evaluation goals. We foresee tree major experimentation/ evaluation opportunities of our approach in practice, where the measures will be collected and analyzed: (i) in a commercial context, applying modules of our infrastructure to improve a current software development process; (ii) in the Demoiselle Process definition and evolution (www.frameworkdemoiselle.gov.br); (iii) in the Qualipso project context (http://www.qualipso.org/). In Qualipso, CDEs are being exploited as a means of managing Open Source Factory knowledge. Recently we are considering that the infrastructure may be useful for documenting and evolving its Open Source process too, following the approach presented here. Qualipso project aims to promote trustiness on open source development, and it is particularly interested in trustworthy elements related to development process.

6. Related works

Distributed software development is not a trivial task and it is becoming common both in national and international organizations. Different solutions have been proposed to deal with its complexity. For instance, some reports were found related to the usage of CDE inside large organizations and to the understanding of networks of communities around the development of software systems. However, their focus is on promoting an environment for developing the software itself, not for supporting the SPI endeavor. To the best of our knowledge most DSD studies focus on developers and their activities not in SPI professionals or SPI activities. Even so, the following experiences on using CDE or fostering software development communities in large organizations were considered and adapted to SPI context: Riehle et al. (2009), Gurbani et al. (2006), Dinkelacker et al. (2001), Melian et al. (2002) , Hupfer et al. (2004).

Vanzin et al (2007) present practices to define a global software process for a distributed environment in a case study. It was useful for our approach as it brings factors that may impact process definition related to distributed development characteristics. Our approach is stronger related to SPI key success factors in addition. We could not find similar proposal of collaborative and distributed SPI strategies to large organizations.

An initial idea of applying a hypermedia tool to monitoring level of the software process management model for distributed groups was introduced by Maindantchick et al.(1999).Existing SPI tools typically support assessors in collecting data during assessments. They provide reporting capabilities to aggregate the collected results. ColabSPI goes in a different/complementary direction and proposes a web-based project workspace to support SPI teams in handling different phases of typical SPI lifecycle. Some works focuses in the content of the distributed development process itself, for instance APSEE-Global (Freitas, 2005), in the context of Process Based Software Engineering Environment, extends the environment to distributed software development characteristics. Our approach focuses on mechanisms to better improve software processes definition, evolution and support.

7. Conclusions

This paper presented an approach that supports geographically distributed SPI initiatives. It is suitable to major SPI models and guidelines and helps many phases of the SPI life cycle. The ColabSPI infrastructure prototype was presented together with some preliminary results. ColabSPI requirements were defined to handle major SPI critical success factors. Currently the evaluation of ColabSPI in three different contexts is being pursued. For more precise results, the GQM paradigm is being considered. The discussion of distributed and collaborative SPI in the Qualipso project context may open new directions for ColabSPI, for instance the maintenance and administration of a maturity model and not only the process.

References

Basili, V. R. Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Technical Report UMIACS-TR-92-96. University of Maryland, 1992.

Basili, V., Caldiera, G. and Rombach, H.D. Experience Factory. In: Encyclopedia of Software Engineering, v. 1, pages 469—476, 1994.

Bjornson, F. Knowledge Management in Software Process Improvement. Doctoral Thesis. Norwegian University of Science and Technology, 2007 (Appendix A).

Booch G., Brown A.W. Collaborative Development Environments. Rational Software Corporation, 2002.

Dinkelacker, J. Garg, P., Miller, R. Nelson, D. Progressive Open Source. Tech ReportHP, 2001.

Florac, W., Park, R., Carleton, A. Practical Software Measurement: Measuring for Process Management and Improvement. Guidebook CMU/SEI-97-HB-003. SEI, PA, USA, 1997.

Fogel, K. Producing Open Source Software. O'Reilly Media. CA, USA. 2006.

Freitas, A.V. *APSEE-Global: um Modelo de Gerência de Processos Distribuídos de Software*. Master Thesis, UFRGS, 2005 (In Portuguese).

Gurbani, V., Garvert, A., Herbsleb, J. A Case Study of a Corporate Open Source Development Model. ICSE'06, May 20-28, 2006, Shanghai, China.

Haumer, P. Eclipse Framework Composer. Available at: http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf. April, 2007.

Hupfer, S., Ross, S., Patterson, J. Introducing collaboration into an application development environment. Proc. of the ACM conference on Computer supported cooperative work, 2004.

ISO/IEC 15504 Information Technology - Software process assessment. 2003.

Maindantchick, C.; Rocha, A. Managing a worldwide software process. In: International Workshop on Global Software Development, ICSE, 2002

Maindantchick, C.; Rocha, A. Xexeo, G. Software process standardization for distributed working groups. In: Proc. 4th IEEE Int. Symp. And Forum on Soft. Eng. Standards, 1999.

Malheiros, V. Cunha, L. Rehem, S. *Mantis-PMP: uma ferramenta livre para gestão de mudanças em processos*. ConSerpro, Brazil, 2007 (In portuguese).

Malheiros, V. Paim, F. Mendonça, M. Continuous Process Improvement at a Large Software Organization. Software Process: Improvement and Practice, Wiley InterScience, 2008a.

Malheiros, V. Rehem, S. Maldonado, J.C. *Atabaque: uma contribuição de sucesso na evolução de processos*. SBQS, 2008b (In Portuguese).

McFeeley, R. IDEAL: A User's Guide for Software Process Improvement. CMU/SEI-96-HB-001, ADA305472. Pittsburgh, PA: SEI, Carnegie Mellon University, 1996.

Melian, C., Ammirati, C, Garg P., Sevón, G. Building Networks of Software Communities in a Large Corporation. Tech Report HP, 2002.

Moe, N., Dyba, T. 2006.The Use of an Electronic Process Guide in a Medium-sized Software Development Company, Software Process Improvement and Practice, Wiley InterScience.

Niazi, M., Wilson, D., and Zowghi, D. A framework for assisting the design of effective software process improvement implementation strategies. J. Syst. Softw. 78, 2. Nov., 2005).

Osterweil, L. Software processes are software too (revised). In Proc. of ICSE, 1997.

Prikladnicki,R., Damian,D. and Audy,J. Patterns of Evolution in the Practice of Distributed Software Development: Quantitative Results from a Systematic Review. Evaluation and Assessment in Software Engineering (EASE), Bari, Italy, 2008.

Prikladnicki, Rl ; Audy, J.; Evaristo, R.. Global Software Development in Practice: Lessons Learned. Software Process Improvement and Practice, USA, v. 8, n. 4, p. 267-281, 2004

Ramasubbu, N. and Balan, R.. Globally Distributed Software Development Project Performance: An Empirical Analysis. ESEC-FSE’07, Croatia, 2007

Rehem, S. *Relatório de avaliação de ferramentas livres para Ambiente de Desenvolvimento Colaborativo.* Tech Report - SERPRO, 2008 (In Portuguese).

Reis, C. Fortes, R. An Overview of the software engineering process and tools in the Mozilla Project. Workshop on Open Source Software Development, 2002.

Riehli, D. et al. Open Collaboration within corporation using software forges. IEEE Software, v.26 n.2, 2009

SEI - Software Engineering Institute. Capability Maturity Model Integration (CMMI SM), Version 1.2. Technical Report CMU/SEI-2002-TR-011, SEI, 2002.

Softex. *MPS.Br - Guia de Avaliação*. 2006. (In Portuguese)

Vanzin, M., Ribeiro, M., Prikladnicki, R., Ceccato, I., Antunes, D. Global Software Processes Definition in a Distributed Environment. In Proc. 29th Annual IEEE/NASA on Software Engineering Workshop. IEEE Computer Society, 2007.