Uma contribuição para a melhoria colaborativa e distribuída de processos de software

Viviane Malheiros

SERVIÇO DE PÓS-GRADUAÇÃO DO
ICMC-USP

Data de Depósito: 05/04/2010

Uma contribuição para a melhoria colaborativa e distribuída de processos de software

Viviane Malheiros

Orientador: Prof. Dr. José Carlos Maldonado

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Cinências - Ciências de Computação e Matemática Computacional.

USP – São Carlos
Abril 2010

Resumo

A área de melhoria de processos de software (MPS) tem sido investigada sistematicamente, dadas as evidências de que a qualidade do processo pode influenciar significativamente na qualidade do produto final. Modelos e guias com boas práticas para a MPS têm sido sintetizados. Ainda assim, a literatura carece de trabalhos que estabeleçam estratégias de como implementar as boas práticas sugeridas por tais modelos e guias na indústria. Em paralelo, o desenvolvimento distribuído de software (DDS) tornou-se uma realidade, aumentando a complexidade e a importância do processo de desenvolvimento de software e demandando estratégias que permitam uma MPS também distribuída. O objetivo deste trabalho é investigar estratégias e mecanismos que possam promover uma MPS distribuída e colaborativa. A ColabSPI, uma estratégia colaborativa e distribuída para MPS, é proposta para apoiar a evolução estruturada do processo; o tratamento de propostas de melhorias de processo; e a comunicação e participação dos desenvolvedores na MPS. Durante a investigação, fatores que podem influenciar a MPS foram identificados e a relação entre eles foi mapeada, tanto a partir da literatura quanto a partir de um estudo em campo. Dois contextos foram explorados durante a investigação: (i) a MPS em uma organização de grande porte, com unidades distribuídas; e (ii) a MPS no desenvolvimento de software livre (SL). Contribuições para a MPS no desenvolvimento de SL foram geradas no contexto do projeto internacional Qualipso, com a co-autoria do Qualipso Open Source Maturity Model (OMM) e a adequação de ColabSPI para evoluir o OMM. Em ambiente industrial, estudos experimentais foram conduzidos para validar a aplicação de ColabSPI e revelaram que algumas práticas do desenvolvimento de software, inclusive do DDS e do desenvolvimento de SL, podem ser aplicadas com sucesso à MPS, trazendo ganhos de eficácia e eficiência para a melhoria de um processo de desenvolvimento de software. A estratégia proposta está sendo base também para a definição do ambiente de MPS do processo Demoiselle, um processo livre para desenvolvimento de software.

Abstract

Software Process Improvement (SPI) has become an active research field, given the evidences that the quality of software processes can significantly influence the final product quality. Therefore, SPI models and guides have been disclosed. Still, there is a lack of studies establishing strategies on how to implement SPI models' best practices. Meanwhile, distributed software development (DSD) is becoming widespread, increasing the complexity and importance of software processes and demanding distributed SPI strategies. This study aims to investigate strategies and mechanisms to promote a distributed and collaborative SPI. ColabSPI, a collaborative and distributed approach to SPI is proposed to supporting process evolution, handling process improvement proposals, and promoting the developers' participation in SPI and communication. Accordingly, during the investigation possible influences to SPI programs were explored and the relationship between them has been mapped, both from the literature and from a field study. SPI was investigated in two contexts: (i) a large organization, counting on distributed development units; and (ii) the FLOSS development environment. Contributions to SPI in the FLOSS development environment took place in the international project Qualipso resulting on a ColabSPI customization to evolve the Qualipso Open Source Maturity Model (OMM) and in the co-authoring of the model. In industrial environment, experimental studies were conducted to validate ColabSPI, revealing that some of the software development practices, including DSD and FLOSS practices, can be successfully applied to SPI and yield a more efficient and effective improvement of the software development process. ColabSPI features are also being considered to the definition of a SPI environment to the Demoiselle process, an open process for software development.

Aos meus pais e irmãs.

A Robert.

Agradecimentos

A meu orientador, Prof. Dr. José Carlos Maldonado pela grande confiança, pelo apoio e pela oportunidade de trabalhar em seu grupo de pesquisa e no projeto Qualipso.

A Robert pelas casinhas de sapê, e por compartilhar a dor e a delícia de ser doutoranda.

A meus pais e irmãs pelo amor incondicional, por sempre estarem por perto. Obrigada pelo exemplo e pelo apoio estratégico-tático-operacional-emocional para esse trabalho.

Aos meus amigos, a cada um de vocês! Em especial, aos mais próximos durante o doutorado: Doda, Lau, Kika, Mari, Lú, Marcos, Martinha. Aos novos sobrinhos!

A minha família pelo incentivo e admiração. A meus avôs Pedro e Aurelino que me viram doutoranda, mas partiram antes de poder ver a neta "dôtora". A minhas avós, a Dinho e aos novos priminhos. A minha *family in-law*. A vó Juju!

To Dr. Carolyn Seaman for opening to me her research group and her house; and for being a friend and an example of professional to be followed; also to the University of Maryland, Baltimore County, for the opportunity to be there.

To my friends Sreedevi, Gunes, Susan and Yuepo for the moments of fun and for sharing their knowledge with this "doctor to be". A particular thanks to Dani, my new norwegian-american-brazilian friend for all the support.

Aos amigos “são carlenses” pelos momentos que passamos juntos. Fabiano, Erika, Alê, Marcos, Ellen, Márcio, Sandro, Otávio, Carol, foi muito bom conviver com vocês.

Aos amigos e colegas de trabalho, entre eles: Serge, Prof. Dr. Manoel Mendonça, Fábio Rilston, Gisela, Gustavo, Eduardo, pelas ricas discussões e troca de idéias, pelo apoio, e pelos trabalhos conjuntos. Aos colegas de Ribeirão, vocês fizeram nossa estada aqui melhor. Obrigada pelo carinho e atenção. Vou sentir saudades!

A Cida, pelo carinho e pela ajuda essencial durante os anos de estudo.

Aos professores e funcionários do ICMC, em especial Graça, Ana, Laura, Tati, Seu Arly, Otávio e Beth, pelo cuidado e presteza.

Ao Skype, MSN, Orkut, Facebook, Sedex, telefone VoIP e celular, por existirem.

Às cidades São Carlos e Columbia, meus vilarejos, onde existe um sonho bom!

Ao SERPRO e à CAPES (programa PDEE) por viabilizarem financeiramente o doutorado. Ao SERPRO também pelo ambiente favorável ao desenvolvimento do conhecimento.

Sumário

1- Introdução....19
1.1 Motivação e contexto....19
1.2 Objetivos....22
1.3 Relevância e aplicabilidade das contribuições....22
1.4 Organização da tese....25
2- Revisão bibliográfica....27
2.1 Considerações iniciais....27
2.2 Processo de desenvolvimento de software....28
2.2.1 Documentação de processos de desenvolvimento de software....29
2.2.2 Métodos ágeis....35
2.3 Melhoria de processo de software (MPS)....37
2.3.1 Guias para implementar a MPS contínua....38
2.3.2 Boas práticas de melhoria de processos no CMMI....41
2.3.3 Um exemplo de modelo de maturidade brasileiro....43
2.3.4 Outras iniciativas de estratégias para a MPS....44
2.4 A gestão de conhecimento (GC) e o desenvolvimento de software....45
2.4.1 GC: visão geral....45
2.4.2 Uso de práticas de GC no desenvolvimento de software....48
2.4.3 Ambientes e ferramentas de GC....53
2.5 Desenvolvimento distribuído e desenvolvimento de software livre....59
2.5.1 Desenvolvimento distribuído de software (DDS)....59
2.5.2 Desenvolvimento de software livre....64
2.5.3 O projeto Qualipso....67
2.5.4 Aplicação de práticas do desenvolvimento de SL em organizações....70
2.6 Considerações finais....71
3- Fatores que influenciam os resultados da MPS....75

3.1 Considerações iniciais....75
3.2 Identificação de fatores de influência para MPS....76
3.3 Analisando relações identificadas entre fatores da MPS....78
3.3.1 Categoria Aspectos Organizacionais....80
3.3.2 Categoria Melhoria Contínua....82
3.3.3 Categoria Aderência....85
3.3.4 Categoria Participação e Motivação....89
3.3.5 Categoria Comunicação....92
3.4 MPS e o mundo do software livre....94
3.5 Considerações finais....95
4- Experiências em MPS e GC aplicadas ao desenvolvimento de software no SERPRO 97
4.1 Considerações iniciais....97
4.2 Experiência em MPS no SERPRO....98
4.2.1 Processo SERPRO de Desenvolvimento de Soluções (PSDS)....98
4.2.2 Programa SERPRO de Melhoria do Desenvolvimento....101
4.2.3 Identificação de melhorias para o PSDS, para a MPS e as lições aprendidas.106
4.3 Experiência de uso de GC no desenvolvimento de software....110
4.4 Considerações finais....112
5- ColabSPI - estratégia colaborativa e distribuída para MPS....115
5.1 Considerações iniciais....115
5.2 A construção da ColabSPI....116
5.3 Princípios....118
5.4 Estrutura organizacional para MPS....118
5.5 Etapas para MPS....124
5.5.1 Preparar para a MPS....126
5.5.2 Refinar os objetivos e definir o direcionamento da MPS....126
5.5.3 Planejar iterações da MPS....127

5.5.4 Desenvolver iteração da MPS 128
5.5.5 Revisar e publicar nova versão do processo.....128
5.5.6 Usar a nova versão do processo 129
5.6 A infraestrutura de MPS 129
5.6.1 Arquitetura da infraestrutura da ColabSPI 132
5.6.2 Documentação do processo 134
5.6.3 Estratégias para tratamento das propostas de melhoria de processo 139
5.6.4 Colaboração e comunicação 145
5.7 Considerações finais 147
6- Estudos de caso de aplicação da ColabSPI 149
6.1 Tipos de estudos experimentais utilizados.....149
6.2 Aplicação da ColabSPI para melhoria de processos na indústria 151
6.2.1 Estudos relacionados à documentação e à evolução estruturada de processos 151
6.2.2 Estudos relacionados ao tratamento de PMPs.....161
6.2.3 Estudos relacionados ao uso de um Ambiente de Desenvolvimento Colaborativo (ADC) para a MPS.....173
6.2.4 Estudos relacionados ao uso de pilotos para MPS 176
6.3 Utilização da estratégia para a evolução do OMM.....179
6.3.1 Organização das comunidades do OMM 180
6.3.2 O ciclo de evolução do OMM 181
6.3.3 A plataforma de evolução do OMM 183
6.3.4 Resultados preliminares da adaptação da ColabSPI para evoluir o OMM 185
6.4 Considerações finais 185
7- Conclusões 187
7.1 Trabalho desenvolvido.....188
7.2 Outras contribuições dentro do trabalho de doutorado 189
7.3 Dificuldades e limitações do trabalho realizado 191

7.4 Perspectivas de trabalhos futuros 192
7.5 Produção científica 194
Referências bibliográficas 197
Apêndice A 211
A1. Exemplo de EAP para gerenciar a MPS 211
A2. Exemplo de política de publicação de processo 212
A3. Exemplos de notificações para o tratamento PMPs 213
A4. Exemplo de Plano para a MPS 214
A5. Passos para implantar a MPS 218

Lista de Figuras

Figura 1: Mapa mental do embasamento teórico utilizado nesta tese 27
Figura 2: Terminologia principal do SPEM 2.0. Adaptado de OMG (2008) 33
Figura 3: Ecossistema do EPF. Extraído de Eclipse (2007) 34
Figura 4: Dimensões para selecionar abordagens. Adaptado de Boehm e Turner (2004) 36
Figura 5: Modelo IDEAL. Extraído de Gremba e Myers (1997) 40
Figura 6: O *Quality Improvement Paradigm*. Extraído de Basili e Caldiera (1985).....41
Figura 7: Áreas de processo básicas - gestão de processos. Adaptado de SEI (2006) 42
Figura 8: Áreas de processo avançadas - gestão de processos. Adaptado de SEI (2006) .42
Figura 9: Espiral do conhecimento de Nonaka e Takeuchi (1997) apud Passos (2004) ...46
Figura 10: Modelo de arquitetura de GC. Adaptado de Lindvall et al. (2002) 53
Figura 11: Modelo de referência para o DDS. Extraído de Prikladnicki (2003).....60
Figura 12: Matriz CSCW de duas dimensões. Adaptado de Wikipedia (2010b) 62
Figura 13: Estrutura do projeto Qualipso. Extraído de Qualipso (2005).....68
Figura 14: Níveis de maturidade do OMM. Adaptado de Wittmann et al.(2009).....69
Figura 15: Principais passos executados para analisar fatores para a MPS.....76
Figura 16: Notação para o esquema de fatores de influência à MPS 78
Figura 17: Diagrama geral de fatores que podem influenciar a MPS 80
Figura 18: Fatores da categoria “Aspectos Organizacionais” 81
Figura 19: Fatores da categoria “Melhoria Contínua”.....82
Figura 20: Fatores da categoria “Aderência” 86
Figura 21: Fatores da categoria “Participação e motivação”.....90
Figura 22: Diagrama de causa e efeito para “Participação e motivação”.....91
Figura 23: Fatores da categoria “Comunicação” 92
Figura 24: Analisando influências para comunicação e colaboração.....93
Figura 25: Exemplo de recomendação para processos de desenvolvimento de SL 95
Figura 26: Tela principal do PSDS (SERPRO, 2007) 100
Figura 27: Estrutura da MPS no SERPRO (MALHEIROS et al., 2008a).....103
Figura 28: Diagrama de causa e efeito de fatores identificados para a MPS do SERPRO 109
Figura 29: Análise de fatores para demora no tratamento de PMP 109
Figura 30: Análise de fatores para demora em esclarecer dúvidas dos usuários.....110
Figura 31: Componentes da estratégia ColabSPI 116

Figura 32: Estrutura organizacional para MPS da ColabSPI 120
Figura 33: Etapas do ciclo da MPS 125
Figura 34: GC na ColabSPI. Adaptado de Lindvall et al. (2002) para a MPS 130
Figura 35: Organização das estratégias da ColabSPI por grupo funcional 132
Figura 36: Arquitetura da infraestrutura da ColabSPI.....133
Figura 37: Contexto da documentação e evolução de processo documentado.....134
Figura 38: Diagrama de estado proposto para tratamento de PMP 141
Figura 39: O espaço virtual para o projeto de MPS 146
Figura 40: Histórico de *download* para os arquivos da Atabaque 152
Figura 41: Evolução dos estudos sobre documentação estruturada de processos 153
Figura 42: Editor visual - alterando a macroatividade "Gestão de Projetos" do PADS..155
Figura 43: *Site* do PADS publicado durante o estudo de viabilidade.....156
Figura 44: Tela do processo Demoiselle gerada com o EPF Composer 160
Figura 45: Evolução dos estudos relacionados ao tratamento de PMPs.....161
Figura 46: Extrato da tela principal da GM-PSDS (MALHEIROS et al., 2006) 162
Figura 47: Atendimento de PMP por macroatividade (MALHEIROS et al., 2006) 164
Figura 48: Atendimento de PSs por macroatividade (MALHEIROS et al., 2006) 165
Figura 49: Tela "Minha Visão" de Mantis-PMP (MALHEIROS et al. 2007b).....168
Figura 50: Diagrama das principais atividades para tratamento de PMPs 170
Figura 51: Diagrama de estado para tratamento de PMP - 1ª. versão da Mantis-PMP ...171
Figura 52: Tela do projeto "processo Demoiselle" no Sourceforge 175
Figura 53: Detalhes do repositório do SVN do processo Demoiselle 175
Figura 54: Detalhe da página de acesso aos arquivos da MPS.....176
Figura 55: Exemplo de MPS com estudo piloto, incorporação de métodos ágeis 177
Figura 56: Comunidades para evoluir o OMM (MALHEIROS e MALDONADO, 2009b) 180
Figura 57: Processo de evolução do OMM (MALHEIROS e MALDONADO, 2009b) 182
Figura 58: Infraestrutura para evoluir o OMM (MALHEIROS e MALDONADO, 2009b) 183
Figura 59: Tela de edição do OMM utilizando a EPF Composer 184
Figura 60: Visão do OMM em formato Web 184
Figura 61: Exemplo de EAP utilizada no planejamento de um programa de MPS.....211
Figura 62: Estratégia para a MPS. Adaptada de Nascimento e Malheiros (2004) 218

Lista de Tabelas

Tabela 1: Taxonomia de estratégias para GC. Adaptado de Earl (2001) 48
Tabela 2: Trabalhos sobre reúso de conhecimento no desenvolvimento de software.....51
Tabela 3: Características marcantes do desenvolvimento de software livre (REIS, 2003b) 66
Tabela 4: Resumo dos principais artigos sobre fatores para a MPS.....77
Tabela 5: Consolidado de recomendações ao PSDS coletadas em visitas às unidades...108
Tabela 6: Contexto da estratégia colaborativa e distribuída para MPS 117
Tabela 7: Requisitos principais para a ColabSPI.....119
Tabela 8: Informações em um plano de comunicação 129
Tabela 9: Necessidades e funcionalidades previstas para a Mantis-PMP 143
Tabela 10: Demais funcionalidades do Mantis-PMP. 144
Tabela 11: Comparação entre os estados de cada fluxo para tratar PMPs 172
Tabela 12: Comparação entre a estrutura para evolução do OMM e do MR-MPS.....182
Tabela 13: Notificações automáticas para o tratamento de PMPs.....213

Lista de Abreviaturas e Siglas

ADC – Ambiente de Desenvolvimento Colaborativo

ADS – Ambiente de Desenvolvimento de Software

CMM - *Capability Maturity Model*

CMMI – *Capability Maturity Model Integration*

CVS – *Concurrent Version System*

DDS – Desenvolvimento Distribuído de Software

EAP – Estrutura Analítica de Projeto

EPF – *Eclipse Process Framework*

GC – Gestão do Conhecimento

GE – Grupo Especialista

GM-PSDS – Gestão de Mudanças do PSDS

GPO – Gestão do Processo da Organização

GQM – *Goal Question Metric*

HTML - *Hyper Text Markup Language*

HTTP - *Hyper Text Transfer Protocol*

IDEAL - *Initiating, Diagnosing, Establishing, Acting, Learning*

KM – *Knowledge Management*

MPS – Melhoria de Processo de Software

MR-MPS - Modelo de Referência para Melhoria de Processos do Software Brasileiro

MPS.BR - Melhoria de Processos do Software Brasileiro

OMM – *Qualipso Open Source Maturity Model*

OPD – *Organizational Process Definition*

OPF - *Organizational Process Focus*

PADS – Processo Atabaque de Desenvolvimento de Soluções

PMP – Proposta(s) de Melhoria de processo

PS- Pedido de suporte

PSDS – Processo SERPRO de Desenvolvimento de Soluções

PSEE - *Process-centered Software Engineering Environment*

RUP – *Rational Unified Process*

SBQS - Simpósio Brasileiro de Qualidade de Software

SGBD – Sistema Gerenciador de Banco de Dados

SEPG – *Software Engineering Process Group*

SERPRO – Serviço Federal de Processamento de Dados

SL – Software Livre

SPEM – *The Software and Systems Process Engineering Meta-model*

SPI – *Software Process Improvement*

SPICE - *Software Process Improvement and Capability dEtermination*

TI – Tecnologia da Informação

UG – Unidade de Gestão

UML – *Unified Model Language*

URL - *Uniform Resource Locator*

WBS – *Work Breakdown Structure*

XMI - *XML Metadata Interchange*

XML – *eXtensible Markup Language*

XSL – *eXtensible Stylesheet Language*

1- Introdução

1.1 Motivação e contexto

Historicamente, a garantia da qualidade de software evoluiu de uma visão que a limitava à qualidade intrínseca do produto final (software sem erro de funcionamento), para um panorama no qual a qualidade do processo de produção influenciaria significativamente na qualidade do software (HUMPHREY, 1989; GOLDENSON e HERBSLEB, 1995; FUGGETTA, 2000; HARTER et al., 2000; ISO/IEC, 2003; GOLDENSON e GIBSON, 2003; PRESSMAN, 2006; SEI, 2006; TRAVASSOS e KALINOWSKI, 2008; SOFTEX, 2009). Alguns autores trazem evidências objetivas que comprovam essa influência (ex.: Harter et al., 2000; Goldenson e Gibson, 2003; Ferreira et al., 2007; Travassos e Kalinowski, 2008). Em particular, o trabalho de Osterweil (1997) estabelece uma analogia entre software e processo de software, uma vez que ambos são executáveis, possuem um conjunto de requisitos, apresentam benefícios se modelados por uma variedade de técnicas e podem evoluir guiados por medições: "*Software processes are software too*".

Para a maior parte das organizações, processos de desenvolvimento de software devem ser tecnologicamente competitivos e adaptáveis e devem ajudar a gerar produtos que atendam consistentemente aos requisitos dos usuários e do negócio. Esta necessidade já era motivo de preocupação na década de 90 (ex.: Florac et al., 1997) e continua sendo discutida em trabalhos mais recentes (ex.: Lindvall et al., 2004; Achatz, 2006). Bons processos devem ajudar a produzir softwares melhores, mais rapidamente e com custo menor. Nesse contexto, a Melhoria de Processos de Software - MPS (*Software Process Improvement* - SPI) é um desafio importante para as organizações.

Diversos avanços foram alcançados para a implantação de modelos e padrões para melhorar o processo de software, como, por exemplo: IDEAL (GREMBA e MYERS, 1997), SPICE (ISO/IEC 15504, 2003), CMMI (SEI, 2006), MR-MPS (SOFTEX, 2009) e a fábrica de experiência (BASILI et al. 1994b). No entanto, a presente lacuna na MPS não é a falta de um modelo ou padrão, mas a falta de uma estratégia efetiva para implementar com sucesso esses

padrões e modelos (NIAZI et al., 2005a). Muita atenção foi dada para *quais* atividades da MPS devem ser implementadas, ao invés de *como* estas atividades devem ser implementadas.

Entender como implementar a MPS com sucesso não é tarefa trivial. A partir da literatura em MPS e observações em campo, foram identificados possíveis fatores que impactam a aderência de projetos ao processo de software e o seu desempenho, no que diz respeito à qualidade e ao tempo. Com base nesse levantamento (capítulos 3 e 4), identificou-se que muitas das influências para o sucesso de programas de MPS estão relacionadas com coordenação, comunicação, colaboração, e, principalmente, com o grau de motivação e participação dos desenvolvedores em iniciativas de MPS.

Mais recentemente, o desenvolvimento distribuído de software (DDS) tem ganhado importância. Existem eventos acadêmicos dedicados especificamente ao tema, como é o caso das conferências internacionais SEAFOOD (*Software Engineering Approaches for Offshore and Outsourced Development*), ICGSE (*International Conference on Global Software Engineering*), e, no Brasil, do WDDS (*Workshop* de Desenvolvimento Distribuído de Software).

Situações em que o desenvolvimento acontece de forma distribuída (equipes de desenvolvimento geograficamente distribuídas) reforçam a necessidade de que as influências para o sucesso da MPS sejam abordadas, uma vez que:

- Processos para o DDS são mais complexos e desafiadores, pois devem lidar com problemas de comunicação e coordenação. De acordo com Maidantchik et al. (1999), tais problemas podem ser reduzidos com o estabelecimento e adoção de um processo bem definido, adequado às especificidades dos ambientes distribuídos. Os efeitos do espalhamento podem ser significativamente mitigados com o uso de processos estruturados de engenharia de software (RAMASUBBU e BALAN, 2007), o que faz do processo de desenvolvimento um fator crítico para DDS (PRIKLADNICKI et al., 2004) e, portanto, torna a MPS extremamente importante no contexto do DDS. Ao mesmo tempo, a melhoria contínua de processos mais complexos é mais difícil; e
- Uma vez que a participação dos desenvolvedores em iniciativas de MPS é um fator crítico para o seu sucesso (NASIR et al., 2008), é importante prover os meios pelos quais desenvolvedores distribuídos geograficamente possam contribuir para melhoria do processo.

Assim, à medida que o DDS torna-se mais comum, a relevância de uma MPS distribuída e colaborativa aumenta.

Tendo isto em mente, esta tese propõe uma estratégia colaborativa e distribuída para a MPS, a ColabSPI, a fim de: (a) melhorar a comunicação e colaboração entre os interessados na MPS; (b) melhorar a participação dos desenvolvedores na melhoria de processos de software; e (c) apoiar a gestão de iniciativas de MPS.

Parte-se da hipótese de que **programas de MPS** podem beneficiar-se de uma estratégia colaborativa e distribuída e de uma infraestrutura, que não só mantêm um repositório de conhecimento sobre o processo de desenvolvimento de software e suas melhorias, mas também permitem aos interessados na MPS gerenciarem-na, comunicarem-se e organizarem o seu trabalho. A ColabSPI pode estimular a emergência e o progresso de um ambiente cooperativo para a MPS. Influências importantes para o sucesso da MPS, como, por exemplo, *troca de conhecimento e apoio a usuários no uso do processo; envolvimento e motivação do pessoal; e comunicação e colaboração* podem ser tratadas.

A proposição da ColabSPI levou em consideração que cada vez mais empresas desenvolvedoras de software disponibilizam os seus modelos de processo em *intranets*, com o objetivo de torná-los mais úteis (MOE e DYBA, 2006). Influências importantes para este trabalho são: (i) DDS; (ii) características do desenvolvimento de software livre; (iii) práticas de gestão do conhecimento; (iii) mecanismos de colaboração, como Wikipédia; e (iv) soluções para monitoramento de erros em software (*Bug tracking system*[1]).

Este trabalho de doutoramento tem foco em grandes organizações que lidam com o DDS e desejam aplicar um processo padrão de desenvolvimento em unidades de desenvolvimento geograficamente distribuídas.

Há evidências também que este trabalho pode ser útil para comunidades de software livre, pois, em essência, o desenvolvimento de software livre dá-se com equipes geograficamente distribuídas, apesar de tipicamente não se utilizar de um processo padrão associado para esse contexto. Explorar o processo como um vetor para aumentar a credibilidade do software livre é o foco do *Qualipso Open Source Maturity Model* – OMM (WITTMANN et al., 2009), parte integrante do projeto internacional Qualipso[2] (*Quality Platform for Open Source Software*). Assim, o conhecimento adquirido com os estudos em MPS foi aplicado no contexto do Qualipso.

[1] *Bug tracking system* é uma aplicação de software projetada para ajudar aos consultores de garantia de qualidade e aos programadores a monitorarem os erros identificados para o seu código.

[2] www.qualipso.org

A aplicação de uma estratégia colaborativa e distribuída para MPS está sendo considerada também para o desenvolvimento do processo Demoiselle. O processo Demoiselle[3] é um processo livre para desenvolvimento de software para o governo com o uso do Framework Demoiselle. O SERPRO é patrocinador do projeto Demoiselle e a autora desta tese é responsável por definir o processo Demoiselle, evoluí-lo e coordenar a sua melhoria.

1.2 Objetivos

Esta tese tem como objetivo principal definir uma estratégia colaborativa e distribuída para melhoria de processos de software (definição e evolução), contemplando o paradigma de desenvolvimento de software livre e princípios de gestão do conhecimento. A estratégia deve apoiar: (i) o tratamento de propostas de melhorias de processo pelas equipes de MPS, (ii) a evolução do processo; (iii) o esclarecimento de dúvidas e a troca de experiências; e (iv) a gestão de programas de MPS.

Para que a aplicação da estratégia seja efetiva é fundamental ter um ambiente (infraestrutura) para apoiar sua implementação. Assim, pretende-se também fornecer subsídios para a construção de um ambiente colaborativo e distribuído de trabalho que seja atraente e efetivo, no sentido de apoiar o tratamento de alguns dos fatores críticos para a MPS em grandes organizações, em especial aquelas que trabalham com DDS.

São objetivos específicos do trabalho:

- Mapear e organizar fatores críticos de sucesso da melhoria de processos de software, identificando a relação entre eles;
- Desenhar uma infraestrutura para apoiar a MPS; e
- Analisar os benefícios e as oportunidades de melhoria para a estratégia proposta em estudos de caso na indústria e no contexto do desenvolvimento de software livre.

1.3 Relevância e aplicabilidade das contribuições

O estabelecimento da MPS foi motivado pela perspectiva de que a qualidade do processo pode influenciar na qualidade do produto. A melhoria contínua do processo pode se provar, muitas vezes, mais complicada e sujeita a falhas do que a definição primária de um processo padrão. Alguns desafios colocam-se nesse contexto:

- Tratar a MPS em organizações de grande porte (formada por várias unidades distribuídas de desenvolvimento de software), garantindo a absorção rápida das

[3] http://sourceforge.net/projects/demoiselle-proc/

mudanças e um processo padrão que atenda a cada uma das várias unidades de software e, ao mesmo tempo, aos interesses corporativos da organização;

- Gerir as informações relacionadas à MPS e comunicá-las para os interessados, incluindo as unidades distribuídas de desenvolvimento;
- Fazer com que o processo seja um agente motivador para a troca e para a geração de conhecimento na instituição ou comunidade;
- Promover o tratamento das melhorias propostas ao processo de desenvolvimento de software sendo evoluído, revertendo-as em otimização, após a sua análise criteriosa; e
- Efetivar as melhorias propostas por meio de evoluções e manutenções no processo.

Esforços no sentido de responder a essas questões possuem ampla aplicabilidade, sendo potencialmente valiosos tanto para organizações que desenvolvem software, e adotam programas de MPS, como para comunidades que desenvolvem software livre. Um terceiro grupo que poderia se beneficiar destas respostas é o das pequenas e médias empresas que se interessem por definir e evoluir conjuntamente um processo de desenvolvimento de software.

Apesar dos benefícios esperados com a melhoria contínua de processos, vários estudos relatam as dificuldades encontradas na tentativa de estabelecer programas de MPS em organizações: Goldenson e Herbsleb (1995); Stelzer e Mellis (1999); Dyba (2001); Baddoo e Hall (2002); Rainer e Hall (2002); Niazi et al. (2005a); e Montoni e Rocha (2007). O assunto tem sido extensivamente discutido em fóruns, grupos e revistas específicos para tratar o tema, como por exemplo: os grupos internacionais *Software and Systems Process Improvement Network*; as revistas *Software Process: Improvement and Practice* e *European Software Process Improvement* e a conferência internacional *International Conference on Software Process*. A crescente distribuição do desenvolvimento de software acrescenta novas influências para o sucesso da MPS, uma vez que um processo padrão de desenvolvimento de software é ainda mais importante nesse contexto (MAIDANTCHIK et al., 1999; PRIKLADNICKI et al., 2004; RAMASUBBU e BALAN, 2007). Maidantchik (1999), por exemplo, propôs um processo padrão específico e um modelo de referência adaptado do CMM - *Capability Maturity Model* (PAULK et al., 1993) para atender às necessidades do desenvolvimento de software geograficamente distribuído.

Em paralelo, o desenvolvimento de software livre tem despertado o interesse e provocado reflexões nos mais diversos âmbitos no Brasil e no exterior: governo, academia e empresas.

O surgimento de uma rede virtual de desenvolvedores e usuários complexa, auto-organizada e com motivações diversas sinalizam a introdução de novas variáveis no setor de

software (SOFTEX, 2005). A forma de organização do trabalho relacionado ao desenvolvimento de software tem suscitado muito interesse entre pesquisadores de diversas áreas, desde o direito até a ciência política e economia (SOFTEX, 2005). A diversidade de áreas interessadas aponta para a relevância do tema. Algumas iniciativas no sentido de entender melhor a dinâmica do desenvolvimento de software livre e seus mecanismos estão surgindo na academia (CUBRANIC e BOOTH, 1999; ASSUNDI, 2001; REIS, 2003b; NAKAGAWA, 2004; SILVA e FALBO, 2006) e em outros setores (QUALIPSO, 2005), mas ainda há muito o que explorar.

No cenário internacional, consórcios e projetos têm surgido para promover o uso de software livre. O OW2[4] e o Qualipso são exemplos de projetos internacionais com esse objetivo.

A motivação do projeto integrado Qualipso é disponibilizar e tornar livre o conhecimento associado ao desenvolvimento e manutenção de software (QUALIPSO, 2005). Seu objetivo é estimular a ampla adoção de software livre (produto e processo), por meio da definição e implementação de tecnologias, procedimentos, leis e políticas que possam ser utilizados pela indústria de desenvolvimento e manutenção de software. Uma estratégia para a MPS colaborativa e distribuída é de interesse para a evolução de modelos e processos livres no contexto do projeto Qualipso. É preciso considerar que a simples divulgação de um modelo ou de um processo livre para desenvolvimento de software pode não ser suficiente para garantir ao usuário a possibilidade de alterá-lo e adaptá-lo, ainda que legalmente ele possua esse direito. Como um processo de desenvolvimento de software deve estar em constante evolução, uma estratégia distribuída e colaborativa para MPS que viabilize a melhoria contínua do processo é de suma importância (MALHEIROS e MALDONADO, 2009b). É importante para a comunidade de usuários que a evolução do processo seja visível para toda a comunidade e dessa forma será mais factível manter a consistência e a unicidade do processo.

No cenário brasileiro, o interesse em SL é crescente também. Já em 2005, o seu impacto na indústria de software brasileira foi mapeado (SOFTEX, 2005). Diferentes órgãos do governo conduzem ações no sentido de promover o uso e o desenvolvimento de software livre, com foco em três temas que são estruturais para o SL[5]: (i) o uso de padrões abertos; (ii) o licenciamento livre dos softwares; e (iii) a formação de comunidades. Uma das experiências, o Portal do Software Público Brasileiro, é um modelo de atuação do Estado para promoção de redes colaborativas e estruturação de um portal de soluções livres. Este portal pretende facilitar a implantação de novas ferramentas; promover a integração entre as unidades federativas; e

[4] www.ow2.org

[5] http://www.softwarelivre.gov.br/comunidade-no-governo, site da comunidade de software livre do governo federal

oferecer um conjunto de serviços públicos para sociedade com base no bem software (http://www.softwarepublico.gov.br/O_que_e_o_SPB). Mais recentemente, foi criado o projeto Demoiselle, cujo objetivo é disponibilizar uma arquitetura de software e um processo de desenvolvimento de software que possam ser livremente utilizados por organizações que desenvolvam software para o governo brasileiro. Um terceiro exemplo de disseminação nacional de um software livre, o Ginga[6], permite a produção de conteúdos audiovisuais digitais (TV digital interativa) interoperáveis e de acesso gratuito aos interessados.

À medida que as comunidades em torno desses projetos crescem e se estruturam, a importância de processos de desenvolvimento confiáveis que apóiem esses projetos cresce também. E, mais uma vez, uma estratégia distribuída e colaborativa para a MPS pode ser utilizada para definição e evolução de tais processos.

Existem evidências de movimento de grandes empresas, em âmbito internacional, no sentido de aproximar-se, adaptar-se e, se possível, apropriar-se do conhecimento e dos processos característicos do software livre. Alguns estudos apontam para uma tendência de aplicar modelos de maior colaboração para a gestão e o desenvolvimento de software (MELIAN et al., 2002; HUPFER et al., 2004) em contextos empresariais. Trabalhos preliminares propõem o repensar da forma de organizar o desenvolvimento de software, incorporando ao ambiente empresarial uma cultura em que há maior colaboração inspirada na concepção de desenvolvimento de software livre (DINKELACKER et al., 2002**;** RIEHLE et al., 2009, GURBANI et al., 2006). Nesse sentido, uma estratégia para melhoria de processos colaborativa e distribuída pode ser muito oportuna para apoiar programas de melhoria de processos de desenvolvimento de software.

Assim, a criação de uma estratégia para MPS colaborativa e distribuída pode beneficiar tanto empresas quanto comunidades de desenvolvimento de software, além de demais agentes do processo.

1.4 Organização da tese

Neste capítulo, foram apresentados: o contexto do trabalho, os principais objetivos da pesquisa conduzida e as motivações para sua realização. O restante do trabalho está organizado em sete capítulos e um apêndice, como descrito a seguir:

- No Capítulo 2, a bibliografia considerada relevante para os objetivos desta tese é apresentada;

[6] http://www.ginga.org.br/

- No Capítulo 3, são discutidas as contribuições desta tese referentes à análise e à organização dos fatores que influenciam a MPS, tomando-se como base uma revisão sistemática na literatura e a experiência na MPS, esta última resumida no Capítulo 4. Os fatores organizados subsidiam o estabelecimento da estratégia para a MPS distribuída e colaborativa. Adicionalmente, fatores que influenciam a MPS para o desenvolvimento de SL também são discutidos, considerando que, cada vez mais, organizações incorporam SL (tanto o software em si, quanto suas boas práticas);
- No Capítulo 4, são caracterizados resultados intermediários desta tese relacionados às experiências em MPS vivenciadas e observadas pela autora nas diversas unidades de desenvolvimento de software do SERPRO - Serviço Federal de Processamento de Dados;
- No Capítulo 5, apresenta-se a ColabSPI, estratégia colaborativa e distribuída para a MPS proposta nesta tese;
- No Capítulo 6, são apresentados estudos experimentais utilizados para análise da viabilidade e da aplicabilidade da estratégia proposta;
- No Capítulo 7, são sintetizados os principais resultados, as suas limitações, e as contribuições do trabalho de doutorado. Perspectivas de pesquisas futuras, decorrentes desta tese, também são discutidas naquele capítulo; e
- Por fim, no Apêndice A, são apresentados exemplos de artefatos, diagramas e informações utilizados em um programa de MPS que servem para ilustrar como pode ser gerenciado um programa de MPS, incluindo um exemplo de política de publicação de processo.

2- Revisão bibliográfica

Neste capítulo, é apresentado o embasamento teórico necessário para a compreensão do trabalho desenvolvido nesta tese. Após as considerações iniciais, os principais referenciais teóricos foram agrupados em quatro grandes áreas: (i) Processos de desenvolvimento de software; (ii) Melhoria de Processo de Software - MPS (*Software Process Improvement* - SPI); (iii) Gestão do Conhecimento - GC (*Knowledge Management* - KM); e (iv) Desenvolvimento Distribuído de Software (DDS) e desenvolvimento de Software Livre (SL).

2.1 Considerações iniciais

Dada a característica multidisciplinar da MPS, uma pluralidade de temas deve ser investigada para respaldar as contribuições resultantes deste trabalho. Na Figura 1, é apresentada uma visão sintética e estruturada dos principais assuntos sobre os quais esse trabalho está fundamentado.

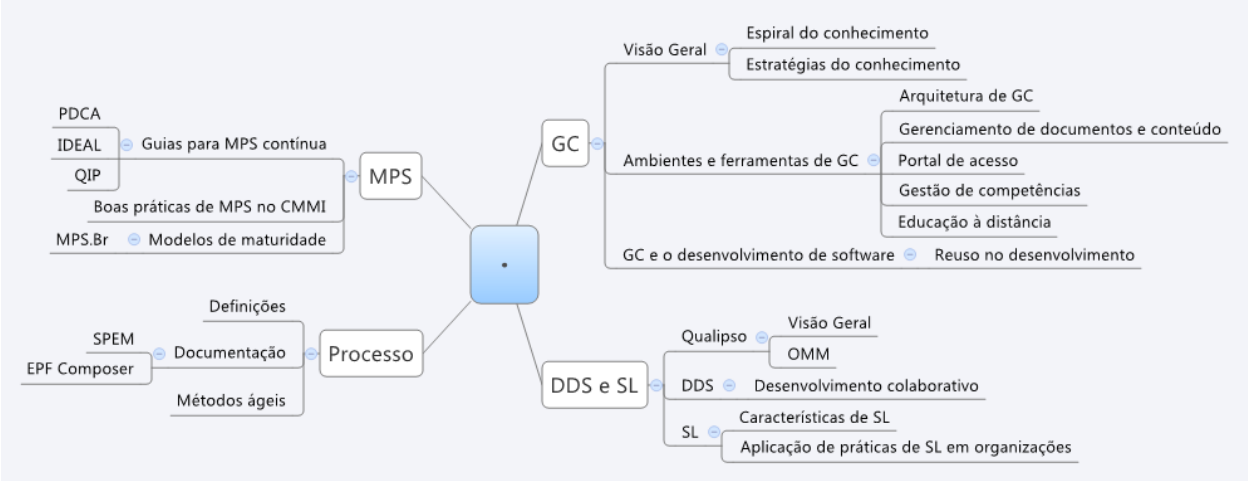


Figura 1: Mapa mental do embasamento teórico utilizado nesta tese

Conceitos de processos de desenvolvimento de software, objeto da melhoria, são apresentados na Seção 2.2. Após a conceituação inicial, aspectos da documentação de processos são abordados. Se adequadamente incorporados a um programa de MPS, o meta-modelo SPEM (*The Software and Systems Process Engineering Meta-Model*) e o EPF Composer podem favorecer a comunicação e o entendimento de um processo e foram considerados para

documentar o OMM (Seção 6.3.3). Assim, ambos são sintetizados. Além disso, metodologias ágeis para desenvolver software têm surgido como uma alternativa a processos tradicionais, e algumas de suas práticas podem ser aplicadas à MPS. Por isso, tais metodologias também são resumidas.

Sendo o foco principal deste trabalho, a Melhoria de Processo de Software (MPS) é apresentada na Seção 2.3. São apresentados modelos de maturidade e guias para implementar a MPS contínua. As áreas de processo específicas para gerência de processos do CMMI também são discutidas, por ser este um dos modelos de maturidade mais difundidos mundialmente.

Considerar aspectos da Gestão de Conhecimentos (GC) é fundamental para guiar o estabelecimento de estratégias de MPS. Uma visão geral da GC e sua aplicação ao desenvolvimento de software são apresentadas na Seção 2.4.

Em seguida, práticas do DDS e do desenvolvimento de SL são apresentadas na Seção 2.5. E, por fim, considerações finais são estabelecidas na Seção 2.6.

2.2 Processo de desenvolvimento de software

Nesta seção, inicialmente, são descritos conceitos relacionados a processo de desenvolvimento de software. Na seqüência, e com base nesses conceitos, são descritos assuntos adjacentes ao tema, como SPEM e métodos ágeis, que são importantes para o desenvolvimento da estratégia proposta (Capítulo 5).

Um processo é uma seqüência de passos para alcançar um dado objetivo (IEEE STANDARD 610.12, 1990). Segundo Jacobson et al. (1999), um processo define quem está fazendo o quê, como e quando para atingir um determinado objetivo. Um **processo de desenvolvimento de software**, segundo Humphrey (1989), é um conjunto de tarefas de engenharia de software necessárias para transformar os requisitos dos usuários em software. Em linhas gerais, neste trabalho, segue-se a definição de Montangero et al. (1999) para processo de software: é a maneira pela qual o desenvolvimento e manutenção de software são organizados, gerenciados, medidos, apoiados e melhorados.

O processo de desenvolvimento de software deve definir: atividades a serem realizadas; recursos necessários; artefatos requeridos e produzidos; procedimentos adotados e o modelo de ciclo de vida adotado (FALBO, 1998). Uma atividade ou tarefa é uma unidade de trabalho que um indivíduo deve executar quando desempenha um papel no desenvolvimento de software. A atividade possui um propósito claro, normalmente expresso em termos de criar ou atualizar algum artefato. A execução de uma atividade envolve papéis e responsáveis e afeta um ou

poucos artefatos. Uma atividade, se razoavelmente complexa, pode ser decomposta em sub-atividades. Os artefatos são produtos de trabalho gerados durante o desenvolvimento e manutenção de software. Modelos, código-fonte, diagramas, planos, unidades de implementação, relatórios ou documentos são tipos de artefatos. As atividades utilizam artefatos como insumo e produzem-nos como resultado de sua execução.

Por traduzir uma forma de trabalho, processos de software normalmente são específicos para determinada organização. Ou seja, diferentes organizações podem utilizar processos diferentes para produzir o mesmo tipo de produto. No entanto, algumas atividades são comuns a todos os processos de desenvolvimento de software (SOMMERVILLE, 2001), por exemplo, identificar requisitos e testar.

A norma internacional ISO/IEC 12207 (*Information Technology – Software Life Cycle Processes*) (ISO, 1995) propõe um modelo de referência de processo e agrupa as atividades típicas de um ciclo de vida de um software desde sua concepção até sua retirada de produção. Esse modelo de referência não representa a implementação de um ciclo de vida ou metodologia, em particular, e por isso pode ser aplicada de diferentes formas dependendo das necessidades da organização. Os processos relacionados ao contexto do sistema de software e ao desenvolvimento do software em si estão organizados em grupos. As atividades de estabelecimento, melhoria e avaliação do processo estão previstas no grupo de processos organizacionais (*Organizational Project-Enabling Processes*), assim como as atividades relacionadas ao estabelecimento e manutenção de uma infraestrutura para o processo.

Alguns trabalhos, inclusive a ISO/IEC 12207 (ISO, 1995), diferenciam processo de desenvolvimento de software de processo de manutenção de software. Nesta tese, no entanto, essa distinção não será considerada, sendo o termo processo de desenvolvimento de software utilizado genericamente e como sinônimo de processo de software, ou seja, englobando também as atividades de manutenção de software.

2.2.1 Documentação de processos de desenvolvimento de software

Segundo Shull et al. (2001), um processo bem definido pode ser observado, medido e conseqüentemente, melhorado. Um processo de software pode estar documentado, ou não. Mas, a documentação do processo é importante para que ele possa ser seguido de forma consistente. Além disso, a documentação pode auxiliar na identificação de pontos de melhoria de processo e no estabelecimento de um processo padrão, comum para todos os projetos de uma organização.

A documentação de um processo é feita por meio de uma representação abstrata de uma ou mais perspectivas desse processo. Essa representação abstrata é também conhecida como

modelo de processo. Os modelos de processo permitem esclarecer as atividades, rotinas, e fatores dinâmicos que determinam a trajetória de um processo de desenvolvimento (CURTIS et al., 1992; DYBA, 2001). Cada modelo de processo o representa, a partir de uma perspectiva em particular, fornecendo uma informação parcial sobre o processo (SOMMERVILLE, 2001). Exemplos tradicionais de modelos de processo de desenvolvimento de software são os modelos cascata e espiral (SOMMERVILLE, 2001). Exemplos mais recentes são RUP - *Rational Unified Process* (IBM, 2010) e os modelos que implementam metodologias ágeis (Seção 2.2.2), a exemplo do *eXtreme Programming* (BECK, 1999). Nos modelos mais recentes, é possível notar uma convergência para processos incrementais e iterativos. Nesses modelos, o desenvolvimento é organizado em iterações com entregas curtas e de valor agregado para o usuário. Tal prática pode trazer como benefícios maior adaptação a mudanças, antecipação de riscos, maior realimentação para o projeto e gerência da complexidade do projeto por meio da decomposição de um problema complexo em problemas menores e gerenciáveis.

Existem diversos formatos (ou padrões ou modelagens) para documentar processos de desenvolvimento de software na literatura, como: ISO/IEC 12207 (ISO/IEC, 1995); RUP (IBM, 2010); SPEM (OMG, 2008); Praxis[7]. No entanto, não foi encontrada uma taxonomia difundida dos elementos que compõem os processos nem sobre como esses elementos podem ser modelados (GENVIGIR et al., 2003 apud CAGNIN, 2005). Máquinas de estado finito, redes de Petri, diagramas de fluxos de dados (CURTIS et al. 1992), diagramas da UML e notação BPMN (OMG, 2005), entre outros, têm sido utilizados para modelar processos de software; variando de um modelo mais descritivo até um modelo automatizado do processo (CURTIS et al. 1992; ARAUJO, 1998; REIS, 2003a). Acuña e Ferré (2010) trazem uma comparação das características de alguns desses modelos. Um dos formatos existentes é o SPEM (*The Software and Systems Process Engineering Meta-Model*), desenvolvido pela OMG (*Object Management Group*), apresentado nesta seção e utilizado pelo EPF Composer. Outro formato é o utilizado pelo RUP, referência para a construção da Atabaque (Seção 5.6.2). Uma terceira alternativa seria documentar o processo de desenvolvimento de software no formato BPMN - *Business Process Modelling Notation*[8]*,* mas nesse caso o foco está na representação do fluxo do processo e não na sua descrição. Não parece haver consenso sobre um formalismo ideal (GIMENES, 2010). Independentemente do formato, um modelo de processo é utilizado para facilitar o entendimento efetivo do processo e é a base para sua execução, gerência e melhoria.

[7] http://homepages.dcc.ufmg.br/~wilson/praxis/
[8] http://www.omg.org/cgi-bin/doc?dtc/09-08-14

Em menor ou maior grau, existe a necessidade de documentar (explicitar) o processo de desenvolvimento de software ou parte dele. As pesquisas na área nos últimos anos têm explorado duas principais vertentes (ARBAOUI et al., 2002 apud BERTOLLO, SEGRINI e FALBO, 2006): (i) abordagens para modelagem, análise e melhoria de processo de software; ou (ii) tecnologia de apoio ao processo de software.

A primeira vertente enfoca abordagens e melhores práticas para estruturação, organização, documentação e descrição de processos de software e inclui normas de qualidade de processo de software e modelos de qualidade de processo (BERTOLLO, SEGRINI e FALBO, 2006), tais como o CMMI (SEI, 2006), o MR-MPS (SOFTEX, 2009), e a norma ISO 15504 (ISO/IEC 15504, 2003). Certamente, esses modelos e normas são referências importantes para se definir como o software será desenvolvido. No entanto, não é objetivo desses modelos prescreverem as ações, as ferramentas, os procedimentos operacionais, as políticas de desenvolvimento e as limitações das organizações de forma precisa (NAKAGAWA, 2006). Essas informações são definidas no processo ou em arcabouços (*frameworks*) de processo. Segundo Fiorini (2001), o mercado fornece alguns *frameworks* de processos que podem ser reutilizados nas empresas. Este é o caso do RUP (IBM, 2010). No contexto de processos livres para desenvolvimento de software, as opções são mais restritas. Iniciativas recentes que podem contribuir para mudar esse cenário são: (i) o Eclipse Process Framework (EPF) (ECLIPSE FOUNDATION, 2005); (ii) o Qualipso (QUALIPSO, 2005); (iii) o Atabaque (MALHEIROS et al., 2008); e o processo Demoiselle. Essas iniciativas são comentadas ao longo deste trabalho (seções: 2.2.1, 2.5.3, 5.6.2 e 6.2.3).

A segunda vertente enfoca desenvolvimento de Ambientes de Desenvolvimento de Software (ADS) Centrado em Processos, integrando ferramentas de apoio ao desenvolvimento de artefatos com ferramentas de apoio à modelagem e execução de processos de software (BERTOLLO, SEGRINI e FALBO, 2006; FREITAS, 2005). Estudos relacionados a essa vertente estão direcionados para a construção de PSEE (*Process-centred Software Engineering Environment*). Um PSEE é um ambiente de desenvolvimento de software no qual os processos utilizados são definidos explicitamente pelo usuário e modelados no ambiente. É composto por uma linguagem para modelar processos e ferramentas para apoiar a definição e execução de modelos de processo (JACCHERI e CONRADI, 1993) e pode facilitar a aquisição de informações sobre o processo. Um PSEE apóia as atividades de definir, monitorar e avaliar um processo para um projeto específico (GIMENES, 2010) e pode ser visto como a automatização de um processo de software (OLIVEIRA, 1999). Exemplos de PSEE são o Spade

(BANDINELLI et al., 1994), o EPOS (JACCHERI e CONRADI, 1993) e, no Brasil, a automatização do processo de desenvolvimento de software nos ambientes instanciados pela Estação TABA[9] (ARAUJO, 1998), o ExAPSEE[10], o APSEE (REIS, 2003a) e mais recentemente o WebAPSEE[11].

Seja baseado em uma ou em outra vertente, atualmente é amplamente reconhecido por pesquisadores que a utilização de métodos, processos e ferramentas de desenvolvimento em projetos pode contribuir decisivamente para a qualidade do produto final. Em ambas as vertentes, uma descrição efetiva dos ativos do processo pode ser decisiva para a compreensão tanto do objetivo de cada componente isoladamente (atividade, papel, etc.) quanto do fluxo a ser adotado (integração dos componentes) (MALHEIROS et al., 2008b). Nesse sentido, documentar e evoluir o processo em um formato mais descritivo pode ser útil inclusive para os ambientes de engenharia de software baseados em processos, que normalmente discutem a evolução de processos no âmbito de melhorias para a sua automatização, mas não destacam as questões relacionadas à evolução da descrição associada ao fluxo do processo automatizado.

Santos et al. (2007) mencionam que a MPS pode ser facilitada por uma infraestrutura de processos de software e se referenciam à Estação TABA como uma opção para ser essa infraestrutura, apresentando resultados positivos da utilização da estação.

SPEM

O meta-modelo de engenharia de processo de software e sistemas (SPEM - *The Software and Systems Process Engineering Meta-model*) é um arcabouço conceitual que pode prover os conceitos necessários para modelar, documentar, apresentar, gerenciar e compartilhar métodos e processos de desenvolvimento (OMG, 2008). O meta-modelo se utilizado para subsidiar a implementação, documentação e evolução de processos pode favorecer a comunicação e entendimento do processo; facilitar a gestão da sua evolução; e facilitar a reutilização de elementos do processo.

Esse meta-modelo apóia a documentação de todos os elementos de processo e a formação de uma base de conhecimentos de tais elementos. Ele provê: uma taxonomia para representação padronizada do processo e seus elementos; uma biblioteca de elementos, que podem ser

[9] A Estação TABA é um ambiente de desenvolvimento de software que apóia a execução das atividades a serem desempenhadas em um processo de software, por meio de um conjunto de ferramentas integradas e repositórios contendo informações adquiridas durante a execução do processo do projeto. Informações sobre a Estação Taba estão disponíveis em http://lens.cos.ufrj.br/es/index2.php?option=com_content&do_pdf=1&id=3).

[10] ExPSEE – Um Ambiente Experimental de Engenharia de Software Orientado a Processo

[11] http://www3.ufpa.br/webapsee/

utilizados em um ou mais processos; e a possibilidade de mapear ciclos de vida de desenvolvimento que representem o fluxo que deve ser seguido. Na Figura 2, é ilustrada a separação entre os elementos de conteúdo do processo (ex.: definição de papéis e artefatos) e os elementos para representação do fluxo (forma como os elementos interagem). Um terceiro grupo de elementos, não representado na Figura 2, é o grupo de visualização (ex.: elemento de configuração), mais relacionado com a apresentação do processo. Todo o arcabouço do SPEM 2.0 está especificado em formato UML (*Unified Model Language*) e pode ser implementado com o apoio da ferramenta EPF Composer (ECLIPSE FOUNDATION, 2007g).

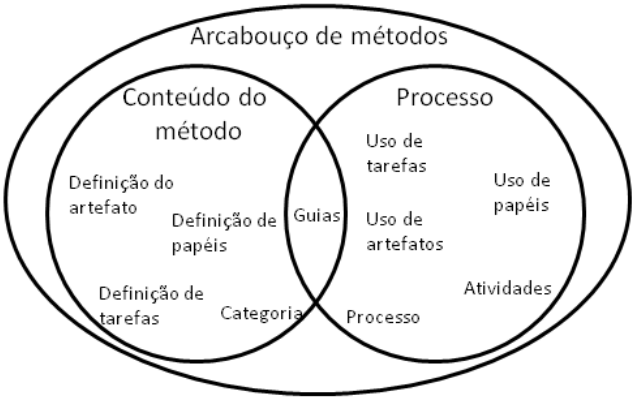


Figura 2: Terminologia principal do SPEM 2.0. Adaptado de OMG (2008)

Eclipse Process Framework – EPF

Dentre o conjunto de projetos associados à plataforma Eclipse[12], o projeto Eclipse Process Framework (ECLIPSE FOUNDATION, 2007d) é especificamente dedicado ao desenvolvimento de um arcabouço extensível para a definição de processos de software (Figura 3). Tendo como ambição o apoio a uma ampla gama de tipos de projetos e de estilos de desenvolvimento, o projeto segue duas linhas de trabalho: (i) o desenvolvimento de um arcabouço extensível para o projeto de processos de software, incluindo o desenvolvimento de algumas ferramentas que exemplifiquem o uso do arcabouço e (ii) alguns exemplos de processos extensíveis que abarquem uma ampla gama de projetos.

O desenvolvimento do EPF Composer Tool (ECLIPSE FOUNDATION, 2007g) segue a primeira linha de trabalho. Esta ferramenta tem como objetivo auxiliar o trabalho de engenheiros de processo, líderes de projeto e gestores de programas na autoria, adaptação e publicação de processos (terceira camada de baixo para cima, Figura 3). O EPF Composer Tool é compatível com o meta-modelo SPEM 2.0.

[12] http://www.eclipse.org/platform/

Na linha de desenvolver processos extensíveis, existem hoje três iniciativas (quarta camada de baixo para cima, Figura 3): (i) o OpenUP/Basic, uma versão de processo minimalista e aberta, inspirada no RUP (IBM, 2010); (ii) o OpenUP/MDD, um componente que contém um conjunto de elementos de processo; e (iii) um componente para o apoio a um processo ágil de desenvolvimento.

O OpenUP/Basic (ECLIPSE FOUNDATION, 2007f) é um processo iterativo de desenvolvimento que busca ser: (i) mínimo, contendo apenas o que é fundamental; (ii) completo, compreendendo todo o processo necessário para o desenvolvimento de um sistema; e (iii) extensível, podendo ser utilizado como base para a adição ou adaptação de conteúdo. Alguns dos seus elementos foram herdados do RUP, após doação da IBM. Estima-se que esta doação represente 15% do RUP (MONTALBANO, 2005).

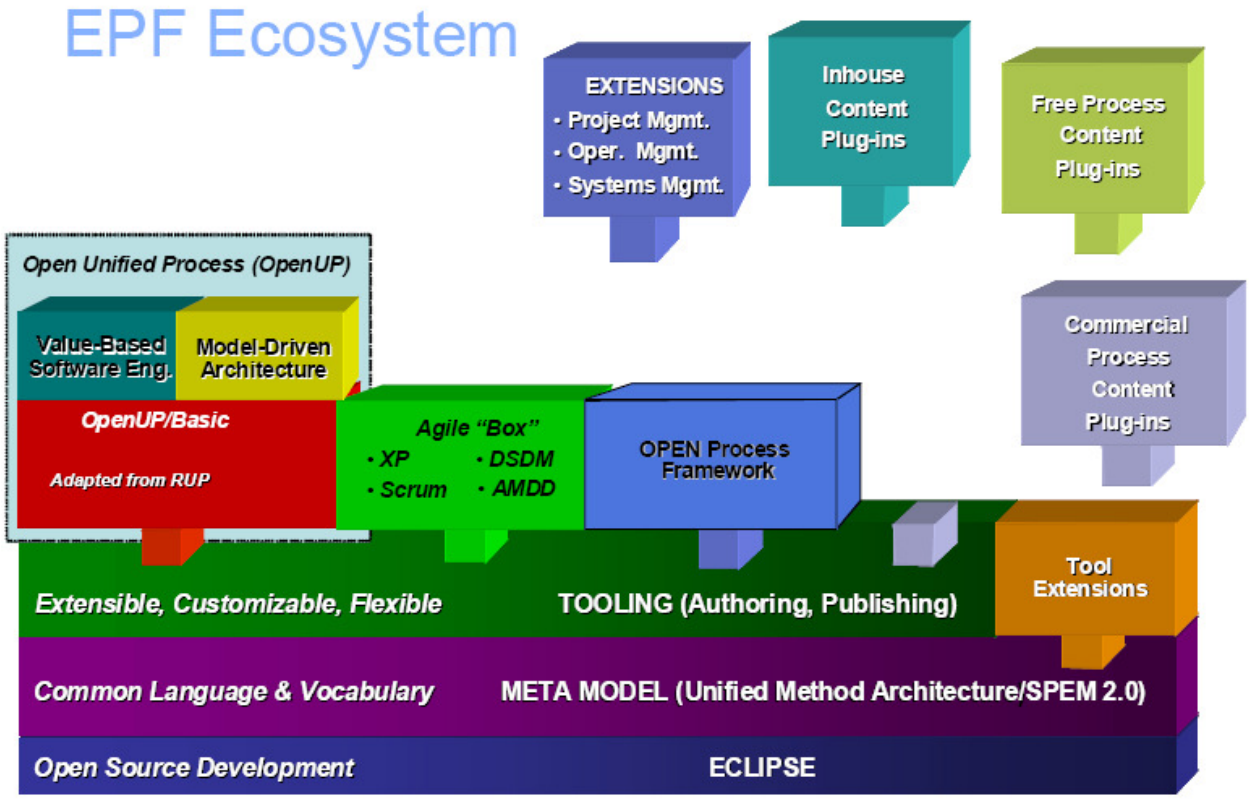


Figura 3: Ecossistema do EPF. Extraído de Eclipse (2007)

O componente OpenUP/MDD tem como propósito fornecer um repositório de elementos de processo que possam ser utilizados no desenvolvimento de processos que seguem uma abordagem direcionada por modelos (*Model Driven Development* – MDD) (ECLIPSE FOUNDATION, 2007e).

Por fim, a última iniciativa objetiva disponibilizar um processo ágil de desenvolvimento como parte do EPF (ECLIPSE FOUNDATION, 2007b). Segundo a Eclipse Foundation (2007b),

contatos estão sendo feitos com a comunidade responsável pelo desenvolvimento de processos ágeis para atingir este objetivo. E, de fato, algumas contribuições começam a surgir.

Para o desenvolvimento desta tese, forma considerados: o SPEM, como vocabulário comum para a definição de processos e o EPF Composer como uma solução para autoria e publicação de processos e, em particular, de modelos.

2.2.2 Métodos ágeis

Diferentes processos podem oferecer diferentes maneiras de organizar o desenvolvimento de software, por exemplo, podem ser mais, ou menos, lastreados em planejamento ou em explicitação do conhecimento. Diversas metodologias ágeis (ABRAHAMSSON et al., 2003) aparecem como uma alternativa a processos mais "tradicionais".

Os valores e princípios das metodologias ágeis estão reunidos no Manifesto Ágil[13]. Esse manifesto é detalhado por meio de doze princípios que sustentam quatro valores estabelecidos: (i) Pessoas e interações são mais importantes que processos e ferramentas; (ii) O software em funcionamento é mais importante que documentação abrangente; (iii) Colaboração com o cliente é mais importante que negociação de contratos; (iv) Responder a mudanças é mais importante que seguir um plano.

Em geral, esses valores determinam que confiar em interações entre pessoas facilita o compartilhamento de informação e mudanças rápidas no projeto quando necessárias. Resumidamente os doze princípios consistem de (CAGNIN, 2005): (i) satisfazer os clientes entregando continuamente versões do software o mais rápido possível; (ii) permitir mudanças de requisitos em qualquer fase do desenvolvimento; (iii) entregar versões do software, que funcionam adequadamente, em curtos períodos de tempo; (iv) proporcionar trabalho conjunto entre desenvolvedores e clientes; (v) fornecer infraestrutura adequada para que indivíduos se tornem motivados e desempenhem o trabalho esperado; (vi) proporcionar comunicação face a face; (vii) alcançar a medida principal de progresso, que é o software operacional; (viii) manter harmonia entre clientes, desenvolvedores e usuários; (ix) preocupar-se com a qualidade técnica e com a execução de bons projetos; (x) projetar com simplicidade; (xi) permitir que as equipes da própria organização participem da definição de arquitetura, requisitos e projeto do sistema; (xii) permitir que as equipes, em intervalos regulares, expressem como podem se tornar mais efetivas.

Os métodos ágeis foram considerados para esse trabalho sob duas perspectivas: (a) inspiraram a definição das estratégias para a MPS e (b) foram aplicados em um estudo de caso de

[13] http://agilemanifesto.org/

melhoria de processo que utilizou pilotos para validar melhorias de processo. Em função de “b”, a pesquisa em métodos ágeis foi fortemente direcionada para relatos de experiência na aplicação de tais métodos em diferentes projetos (DYBA e DINGSOYR, 2008), em particular trabalhos que comparassem métodos ágeis com metodologias mais tradicionais. (LINDVALL et al., 2004; BOEHM e TURNER, 2004; PAULK, 2002; BECK e BOEHM, 2003; ERDOGMUS, 2008).

Alguns autores defendem que métodos ágeis e processos mais tradicionais não são incompatíveis: Boehm e Turner (2004), Larman e Basili (2003), Paulk (2002). Parece adequada a visão de Boehm e Turner (2004) de que uma combinação entre as duas abordagens pode ser mais realista e benéfica. Além disso, como Lindvall (2004) destaca, incorporar métodos ágeis ao desenvolvimento de software em grandes organizações não é uma tarefa simples. Cohn e Ford (2003) discutem algumas abordagens para a transição de processos tradicionais para processos ágeis. Eles reportam algumas questões relacionadas ao desenvolvimento de equipes e, em complemento, relacionadas à formação de departamentos e gestão.

De acordo com Boehm e Turner (2004), a adequação de um ou outro processo depende das características do projeto e do ambiente de desenvolvimento. Eles desenvolveram um gráfico que relaciona cinco principais características que podem influenciar o uso de uma abordagem ou de outra: pessoas, estabilidade dos requisitos, cultura, tamanho e criticidade (Figura 4). Os cinco eixos representam os fatores para diferenciar entre a escolha de métodos mais ágeis (no sentido do centro) e métodos mais planejados (no sentido das extremidades).

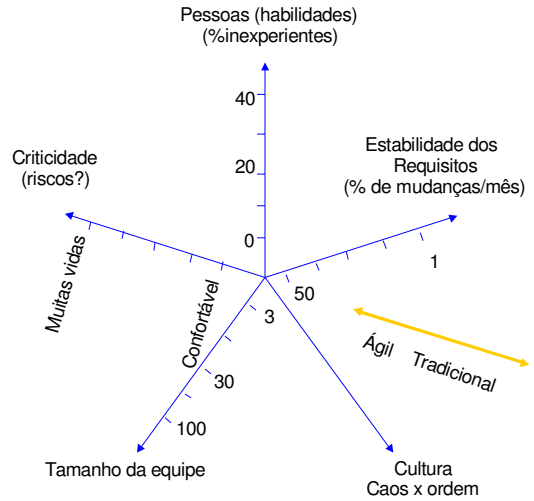


Figura 4: Dimensões para selecionar abordagens. Adaptado de Boehm e Turner (2004)

Outra perspectiva para diferenciar entre as duas abordagens é considerar as características de criação, conversão e transferência de conhecimento (CHAU et al., 2003). Abordagens mais "taylorísticas" apóiam o compartilhamento do conhecimento por meio de sua explicitação.

Abordagens de desenvolvimento mais ágeis se baseiam em socialização por meio da comunicação e colaboração para acessar e compartilhar conhecimento tácito entre os envolvidos.

2.3 Melhoria de processo de software (MPS)

A **melhoria de processo de software (MPS)** é uma disciplina que trata da necessidade de gerir, melhorar e apoiar o uso dos processos de desenvolvimento de software de uma organização (AMBLER, 2005). Por meio da MPS é possível ter o entendimento do processo existente e das mudanças realizadas nesse processo, de modo a melhorar a qualidade do produto de software e/ou reduzir custos e o tempo de desenvolvimento (SOMMERVILLE, 2001).

Dyba (2001) cita três tipos de intervenções relacionadas a mudanças em processos: i) avaliação de processo (*process assessment*) - uma avaliação ou revisão de processo de software sem que ele seja alterado; (ii) melhoria de processo (*process improvement*) - mudanças contínuas ou incrementais no processo de software; e (iii) inovação de processo (*process innovation*) - mudanças radicais no processo, envolvendo substituição total ou parcial do processo. Neste trabalho os três tipos de intervenção são genericamente referenciados como Melhoria de Processos de Software (MPS), como foi feito por Conradi e Fuggetta (2002).

Alguns modelos de maturidade e normas tratam a MPS, compilando boas práticas de desenvolvimento, por exemplo: CMMI (SEI, 2006), ISO/IEC 15504 (ISO/IEC 15504-2, 2003), MR-MPS (SOFTEX, 2009). Os **modelos de maturidade** sistematizam e representam melhores práticas, sugerem medições para avaliação de processos ou podem ser utilizados como um roteiro para melhoria de processos. Por esse motivo, esses modelos são um excelente ponto de partida para definir e evoluir um processo de software. Esse tipo de evolução, baseada em um modelo de referência, pode ser denominado *benchmark* em relação a melhores práticas (DYBA, 2001). Boa parte dos modelos de referência baseia-se nos conceitos de maturidade e capacidade do processo de desenvolvimento de software. Entende-se por capacidade do processo a habilidade deste processo em produzir resultados planejados (PAULK et al., 1993).

Para ser considerada madura pelo SEI - *Software Engineering Institute* (2006), uma organização necessita, além de definir e utilizar, melhorar o seu processo de desenvolvimento de software continuamente. A maturidade de um processo está relacionada com sua estrutura e com controle. O estabelecimento de um processo padrão para a organização e a gestão de um programa de melhoria de processo estão previstos no Nível 3 do CMMI, por meio das áreas de processo: *Organizational Process Definition* (OPD) e *Organizational Process Focus* (OPF). Ambas, mas em especial OPF, reúnem práticas relacionadas à melhoria contínua do processo

organizacional. Essas práticas destacam a necessidade de compreensão e tratamento dos pontos fortes e oportunidades de melhoria identificadas para o processo. Uma explicação sucinta das duas áreas de processo é dada na Seção 2.3.2.

Um modelo de MPS fornece orientação para que a melhoria de processo aconteça pela mudança, atualização, ou aperfeiçoamento dos processos existentes, com base nos resultados de uma avaliação (WANG e KING, 2000). Essa avaliação revelará a capacidade dos processos da organização, indicando as áreas prioritárias para a melhoria e fornecendo subsídios para o planejamento de ações. Por esse motivo, é comum existirem, associados aos modelos de maturidade, **modelos de avaliação**. Estes modelos avaliam se as práticas melhoram a qualidade do software e como o processo de desenvolvimento de software afeta a qualidade dos produtos. São exemplos de modelos de avaliação: o SCAMPI (SCAMPI UPGRADE TEAM, 2006), o MPS.BR - Guia de Avaliação (SOFTEX, 2006) e o SPICE - *Software Process Improvement and Capability dEtermination* (ISO/IEC 15504-5, 2003).

Para alguns autores, modelos de MPS têm demonstrado, na prática, ser uma abordagem viável e efetiva para a melhoria do processo das organizações desenvolvedoras de software (EL-EMAM et al., 1999, 2001; SALVIANO, 2004; TRAVASSOS e KALINOWSKI, 2008). Para outros, entretanto, esses modelos, no formato atual, são incapazes de atacar os desafios mais críticos de uma organização que desenvolve software, como questões culturais ou motivacionais (CONRADI e FUGGETTA, 2002). Iniciativas de MPS podem apresentar baixos níveis de aprovação e sucesso limitado. Segundo Niazi (2006) a taxa de fracasso de iniciativas SPI é estimada em 70%.

A adoção de um modelo de maturidade como referência para melhorar o processo, de fato, não é uma tarefa trivial. Um problema recorrente em melhoria de processo é que o investimento do dinheiro é feito agora, enquanto que o retorno só virá no futuro (CONRADI e FUGGETTA, 2002). Outra questão é a carência de estudos que enfoquem "como" conduzir a MPS (Seção 2.3.4) e não apenas o quê deve ser feito (NIAZI et al. 2005a). Além disso, a MPS engloba variados fatores que dizem respeito à cultura, à tecnologia e aos fatores organizacionais. Uma das contribuições dessa tese é identificar, organizar e co-relacionar os diversos fatores que podem influenciar esforços em MPS. Discussões nesse sentido estão disponíveis no Capítulo 3.

2.3.1 Guias para implementar a MPS contínua

Algumas abordagens sistemáticas para gerenciar o ciclo de vida da MPS foram propostas na literatura: IDEAL *Initiating, Diagnosing, Establishing, Acting, Learning* (GREMBA e MYERS, 1997) e QIP - *Quality Improvement Paradigm* (BASILI e CALDIERA, 1995). Essas

abordagens têm em comum a repetição de um conjunto de atividades (ciclos), com o objetivo de aperfeiçoar o processo a cada novo ciclo. Tais abordagens, apresentadas brevemente a seguir como **guias para** implantar **a MPS contínua**, foram consideradas na proposição da estratégia colaborativa e distribuída para MPS, a ColabSPI.

Modelo PDCA

O ciclo do PDCA foi criado por Walter Shewart e divulgado por Deming (1986). O princípio do PDCA é aplicar um ciclo de melhoria de processo. A aplicação do ciclo produz um processo diferente, a partir de modificações no processo anterior. Este processo modificado é, então, aplicado para gerar produtos, e o seu grau de melhoria é avaliado.

As quatro fases do ciclo PDCA inspiraram outros modelos de melhoria contínua: A) Planejar (*Plan*): estabelecer um plano de melhoria dos processos existentes, definindo metas e métodos para atingi-las; B) Fazer (*Do*): Executar o trabalho planejado e coletar os dados de sua execução; C) Verificar (*Check*): comparar o resultado do trabalho em cada estágio de execução contra os critérios e metas estabelecidos na fase Planejar; e D) Agir (*Action*): estudar os resultados obtidos, identificando o que foi aprendido, quais os problemas encontrados e quais as melhorias para o próximo ciclo, de forma a tomar ações para correção de rumo.

Modelo IDEAL

O IDEAL (GREMBA e MYERS, 1997) é um modelo de ciclo de vida para a melhoria de processo de software, definido pelo SEI e baseado no CMM - *Capability Maturity Model* (PAULK et al., 1993). A dinâmica do modelo é ilustrada na Figura 5.

O nome do modelo expressa as cinco fases que ele descreve (GREMBA e MYERS, 1997 apud NASCIMENTO, 2006):

• **I** - Início (*Initiating*): montar a base necessária para empreender um esforço de melhoria de sucesso, incluindo a definição do objetivo e a infraestrutura para alcançá-lo;

• **D** - Diagnóstico (*Diagnosing*): determinar a situação atual em relação à situação desejada, com base no objetivo almejado e identificar oportunidades de melhoria;

• **E** - Planejamento (*Establishing*): planejar os passos necessários para alcançar o objetivo, estabelecendo prioridades;

• **A** - Ação (*Acting*): realizar os passos especificados de acordo com o planejamento, pesquisando soluções para os problemas e implantando a melhoria em toda a organização; e

• **L** - Aprendizado (*Learning*): aprender com a experiência e melhorar a habilidade de incorporar mudanças futuras.

Figura 5: Modelo IDEAL. Extraído de Gremba e Myers (1997)

Modelo QIP

O QIP (BASILI e CALDIERA, 1995) - *Quality Improvement Paradigm* - define dois ciclos para a melhoria da qualidade do software: o aprendizado do projeto (*project learning*) e aprendizado da organização (*corporate learning*). O ciclo de aprendizado do projeto foca no controle e na execução bem sucedida do projeto de software. Sua operação provê realimentação para o processo, viabilizando a melhoria do mesmo, por meio da análise de resultados coletados. O ciclo de aprendizado da organização, por sua vez, enfatiza o aprendizado, empacotamento de experiências e planejamento de novos projetos. Ele é mais abrangente, no sentido de manipular o conhecimento de diferentes grupos de trabalho, de diferentes execuções do ciclo de aprendizado do projeto. A combinação dos dois ciclos compõe o QIP, esquematizado na Figura 6.

Inspirados no QIP, Mendonça et al. (2008) propuseram um arcabouço (*Framework for Improving the Replication of Experiments* - FIRE) para coordenar e evoluir experimentos controlados em Engenharia de Software. Esse *framework* modifica o QIP em alguns pontos, adaptando-o às características de experimentos, que podem ser diferentes das características de projetos de software.

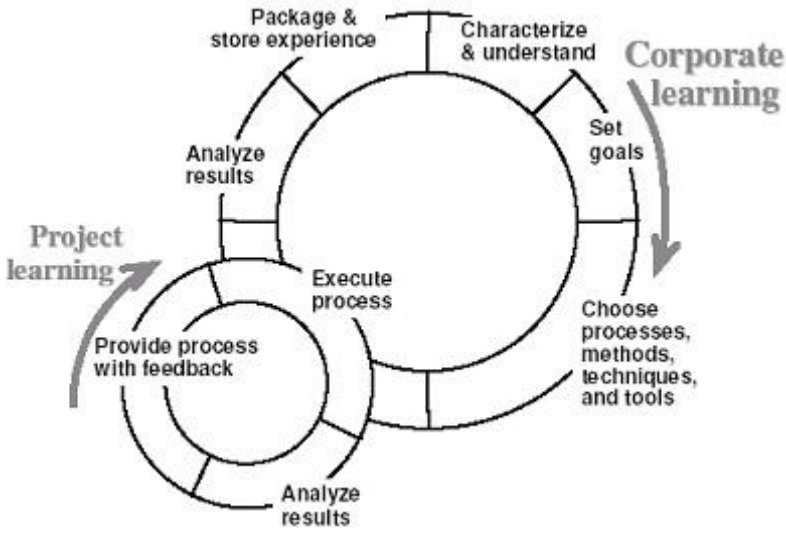


Figura 6: O *Quality Improvement Paradigm*. Extraído de Basili e Caldiera (1985)

2.3.2 Boas práticas de melhoria de processos no CMMI

O gerenciamento de processos de software é a aplicação de conceitos, técnicas e práticas da engenharia de processo para definição, planejamento, distribuição de recursos, aplicação, implementação, monitoramento, controle, avaliação, medição e melhoria de processos (SEI, 2006). Diversos modelos de maturidade fazem referência ao gerenciamento de processos e a MPS. Nesta seção, as áreas de processo do CMMI para gerenciamento de processos serão detalhadas, por ser este um dos modelos de maturidade mais difundidos mundialmente e já ter sido utilizado, referenciado e avaliado em diversos trabalhos. As áreas de processo do CMMI são agrupadas em quatro categorias: (i) Gerenciamento de Processo (*Process Management*); (ii) Gerência de Projetos (*Project Management*); (iii) Engenharia (*Engineering*); e (iv) Suporte (*Support*). As áreas de processo estritamente ligadas à definição e melhoria de processos estão agrupadas na categoria "Gerenciamento de Processo". Essas áreas fornecem uma base para que organizações documentem e compartilhem ativos de processo, boas práticas e lições aprendidas.

Para o CMMI, uma organização madura melhora continuamente seus processos, utilizando como base medição e técnicas de controle estatístico dos processos. A maturidade no CMMI só é alcançada a partir dos níveis 4 e 5. Os níveis 2 e 3 objetivam estruturar a organização para que estratégia de melhoria contínua possa ser implementada nos níveis seguintes.

O início da gestão formal da MPS e o estabelecimento do processo padrão estão previstos para ocorrer no Nível 3, com a implantação das áreas de processo Definição do Processo da Organização (OPD - *Organizational Process Definition*) e Foco no Processo da Organização (OPF - *Organizational Process Focus*). As necessidades de treinamento também são tratadas em nível de organização (OT - *Organizational Training*). A partir da versão 1.2 do CMMI, as

práticas de IPPD (*Integrated Product and Process Development*) relacionadas a processos foram adicionadas à área OPD. Na Figura 7, é ilustrada a interação entre essas áreas de processo.

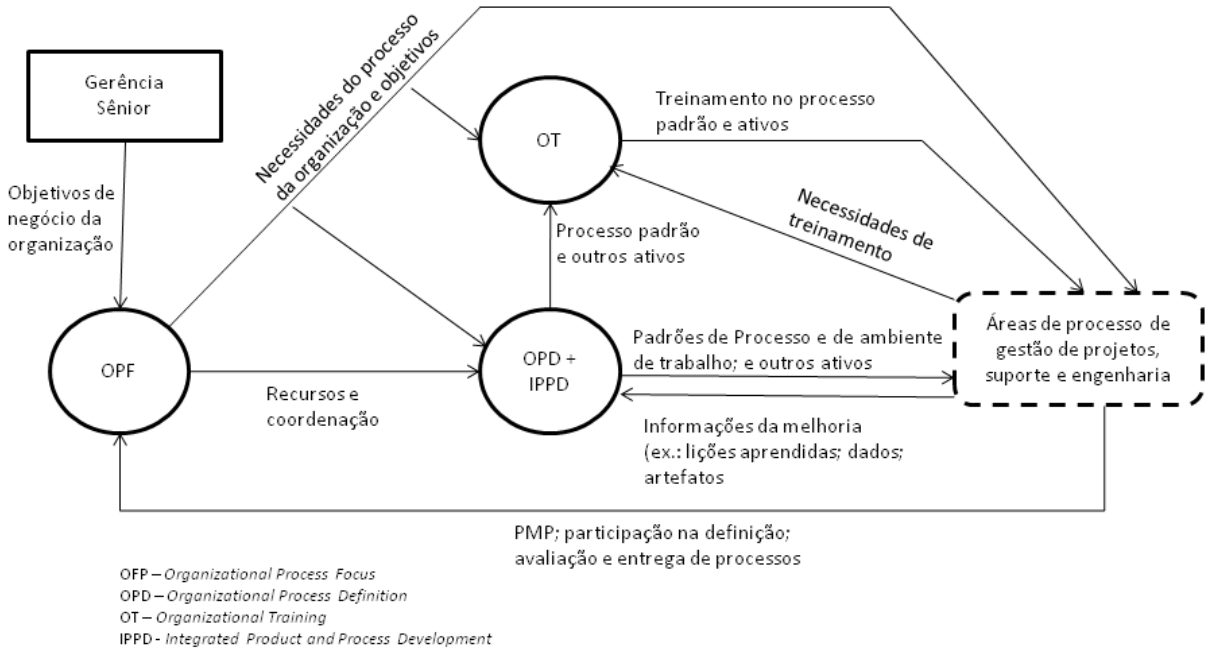


Figura 7: Áreas de processo básicas - gestão de processos. Adaptado de SEI (2006)

Uma vez estabelecido o processo e iniciada a gestão da sua melhoria, o uso das medições para apoiar essa melhoria começa a ocorrer, com a área de processo Desempenho do Processo da Organização (OPP - *Organizational Process Performance*), Nível 4. O programa de melhoria é incorporado ao processo padrão da organização no Nível 5, com a área de processo Inovação e Implantação Organizacional (OID - *Organizational Innovation and Deployment*) (Figura 8).

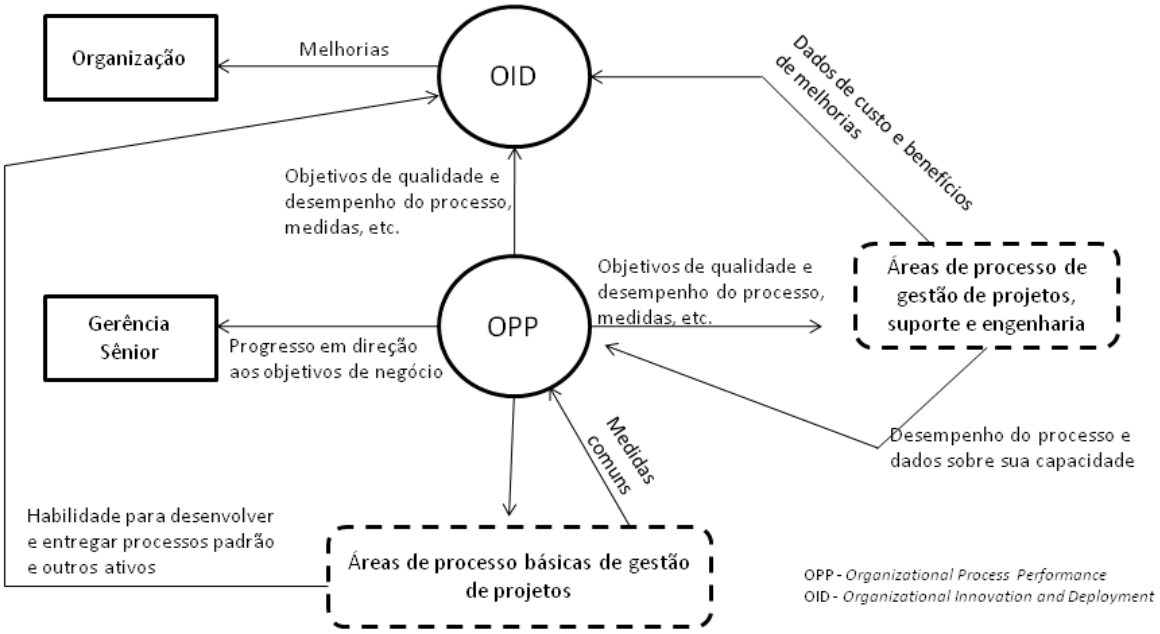


Figura 8: Áreas de processo avançadas - gestão de processos. Adaptado de SEI (2006)

Modelos de referência, como o CMMI, refletem a preocupação de seus autores com questões organizacionais relacionadas à necessidade de: (i) estabelecer diretrizes; (ii) definir evolução e adaptação de processo padrão; (iii) treinar e acompanhar os objetivos estratégicos da organização. No entanto, não é objetivo do CMMI definir um modelo específico do ciclo de vida, nem de estrutura organizacional, nem qual a melhor técnica ou mecanismo a ser utilizado para que a maturidade de uma organização seja alcançada. Essas questões cabem à organização decidir, de acordo com seu contexto e objetivos de negócio.

2.3.3 Um exemplo de modelo de maturidade brasileiro

O objetivo do programa MPS.BR (acrônimo) é a Melhoria de Processo do Software Brasileiro, com duas metas a alcançar a médio e longo prazo (SOFTEX, 2009): (i) meta técnica, visando à criação e ao aprimoramento do modelo MPS; (ii) meta de mercado, visando à disseminação e adoção do modelo MPS, em todas as regiões do país.

Um ponto de partida importante para definir o MPS.BR foi a análise da realidade das empresas brasileiras. O programa de sustentação do modelo foi criado em dezembro de 2003 e é coordenado pela Associação para Promoção da Excelência do Software Brasileiro (Softex). O MPS.BR é uma alternativa brasileira ao modelo norte-americano CMMI. Além de inspirado no CMMI, o MPS.BR é significativamente inspirado nas normas ISO/IEC 15504 (ISO/IEC, 2003) e na norma ISO/IEC 12207 (ISO/IEC, 1995).

O modelo MPS possui três componentes: Modelo de Referência (MR-MPS), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS).

No MR-MPS, a maturidade de processo está organizada em duas dimensões: a **dimensão capacidade** (*capability dimension*) e a **dimensão processo** (*process dimension*). A dimensão capacidade é um conjunto de atributos de um processo que estabelece o grau de refinamento e institucionalização com que o processo é executado na organização. À medida que ganha maior capacidade de desempenhar o processo de desenvolvimento, a organização evolui nos níveis do modelo. Os níveis estabelecem uma maneira racional para aprimorar a capacidade dos processos. A dimensão processo é baseada na ISO/IEC 12207 (ISO/IEC, 1995) e estabelece o que a organização deve executar, para ter qualidade na produção, fornecimento, aquisição e operação de software. A interseção dessas duas dimensões define a maturidade do processo que, no MR-MPS, são sete níveis: A - Em Otimização; B - Gerenciado Quantitativamente; C – Definido; D - Largamente Definido; E - Parcialmente Definido, F - Gerenciado; e G - Parcialmente Gerenciado. Para cada nível de maturidade, áreas de processo foram atribuídas, inspiradas nos níveis 2, 3, 4 e 5 do CMMI em estágios. Esta divisão tem uma gradação diferente do CMMI,

visando a possibilitar uma implementação mais gradual e adequada às micro, pequenas e médias empresas brasileiras. A possibilidade de avaliar considerando mais níveis permite uma visibilidade dos resultados da MPS, na empresa e no país, em prazos mais curtos.

2.3.4 Outras iniciativas de estratégias para a MPS

Durante o desenvolvimento deste trabalho de doutoramento, foram identificadas outras iniciativas para estabelecer estratégias para a MPS: Santos et al., 2007 e Montoni et al., 2008.

Santos et al. (2007) propõem uma estratégia para implantar a MPS, SPI-KM, que se apóia em práticas de GC. Essa estratégia consiste de conjunto de fases definidas que se concentram em questões específicas da implantação da MPS: *Diagnosis, SPI Planning, Process Definition, Training, Mentoring, Knowledge Acquisition, Acquisition and Assessment of Process, Improvement Recommendations, Preparing the Organization for the Appraisal, Processes Assessment*.

A ColabSPI proposta nesta tese apresenta etapas para evoluir um processo de software. Genericamente, as etapas da ColabSPI são compatíveis com as fases da estratégia de Santos et al. (2007), apesar da ColabSPI estar mais focada no planejamento e na gestão da evolução do processo do que em outras fases da MPS. As fases da estratégia de Santos et al. (2007) são compatíveis com o modelo IDEAL (ver detalhe na Figura 5), acrescidas de práticas de GC. Como o modelo IDEAL e as práticas de GC foram insumo para definir as etapas de ColabSPI há afinidade entre ambas. No entanto, aparentemente, o nível de abstração da SPI-KM está mais próximo da estratégia para implantação da MPS proposta por NASCIMENTO e MALHEIROS (2004), cujos principais passos são ilustrados na Figura 62, Apêndice 1.

As estratégias podem se complementar. A ColabSPI, apesar de abordar todo o ciclo da MPS, não enfatiza atividades relacionadas à avaliação do processo e nem enfatiza aspectos como lições aprendidas da MPS, treinamento e consultoria, discutidas por Santos et al. (2007) ou *benchmarking* de outros projetos de MPS (MONTONI et al., 2008). Por outro lado, a ColabSPI oferece maior apoio a questões relacionadas à evolução do processo em si, como evoluí-lo estruturadamente e como tratar as propostas de melhoria de processo, além de tratar questões relacionadas à MPS distribuída e colaborativa e à organização de pessoas para conduzir e executar a MPS.

2.4 A gestão de conhecimento (GC) e o desenvolvimento de software

Segundo Lindvall et al. (2002), um estudo revelou que desenvolvedores de software desperdiçam 40% do seu tempo buscando por diferentes tipos de informações relacionadas aos seus projetos. Com isso, a capacidade de gerir e reter o conhecimento torna-se um dos fatores mais importantes do desenvolvimento de soluções. Nesta seção, são registradas as tendências de uso de gestão de conhecimento para a engenharia de software. Inicialmente, é apresentada uma visão geral da GC, que será útil para compreender a seção seguinte que a relaciona com o desenvolvimento de software. Um panorama do ambiente e ferramentas de GC, com ênfase em sua utilização para a engenharia de software, também são apresentados. A consciência da existência dos diferentes níveis de serviço de GC e ferramentas associadas foram importantes para a proposição da ColabSPI. Na Seção 2.4.3, alguns serviços e ferramentas são ilustrados no contexto do desenvolvimento de software.

Para obter um conhecimento aprofundado das diferentes dimensões de GC relacionadas ao desenvolvimento de software, além da pesquisa abrangente de estudos na interseção dessas duas áreas de pesquisa, foi feita uma pesquisa aprofundada especificamente sobre a aplicação de GC à área de Teste de Software. Esse trabalho não está diretamente relacionado aos trabalhos de melhoria de processos aqui apresentados e, por isso, não está disponível no corpo dessa tese. Os resultados da revisão sistemática realizada serão objeto de um artigo atualmente em elaboração.

2.4.1 GC: visão geral

O conhecimento é um fator que sempre esteve presente na história da humanidade, mas que passou a ter uma importância consideravelmente ampliada em relação a outros fatores econômicos, tornando-se um bem extremamente valioso. Ter controle e facilidade de acesso ao conhecimento tempestivamente passou, há algum tempo, a ser um diferencial competitivo para as organizações.

O conhecimento é gerado a partir da interpretação e reflexão sobre determinada informação, de acordo com um contexto e experiências passadas. Para Nonaka e Takeuchi (1997), existem dois tipos de conhecimento: tácito e explícito. O conhecimento **tácito** depende de experiência pessoal, sendo formado dentro de um contexto social e individual, que envolve crenças e valores. Envolve fatores intangíveis como, por exemplo, *insights*, intuições e emoções. O conhecimento **explícito**, por sua vez, é representado por algum artefato, como um documento

ou um vídeo, que tipicamente foi criado para comunicação com outra pessoa (MARWICK, 2001) e representa uma explicitação de um conhecimento tácito. O conhecimento explícito pode ser expresso em palavras e números e pode ser transmitido e compartilhado.

As formas de interação entre o conhecimento tácito e explícito e entre a organização e o indivíduo resultam em quatro processos de conversão de conhecimento (Figura 9) que constituem a espiral do conhecimento, proposta por Nonaka e Takeuchi (1997):

• Socialização - troca de experiências e, portanto, de criação de conhecimento tácito, como modelos mentais. O aprendizado se dá por meio da observação, imitação e prática. Sem alguma forma de experiência compartilhada, é extremamente difícil para um indivíduo projetar-se no processo de raciocínio do outro.

• Externalização - articulação do conhecimento tácito em conhecimento explícito, ou seja, conceitos, hipóteses, analogias ou metáforas, que podem ser expressos em artefatos. A externalização é considerada a chave para a criação do conhecimento, pois cria conceitos novos e explícitos a partir do conhecimento tácito.

• Combinação - junção de partes de conhecimentos explícitos em um novo conhecimento explícito por meio de análise e reestruturação de informações. Os indivíduos combinam conhecimentos por meio de documentos, reuniões ou conversas informais.

• Internalização - absorção de conhecimento explícito por meio do aprendizado. Aos conhecimentos explícitos (registrados nos artefatos) são adicionados novos conhecimentos tácitos (memórias, intuições, valores) por meio de diversos tipos de experiências.

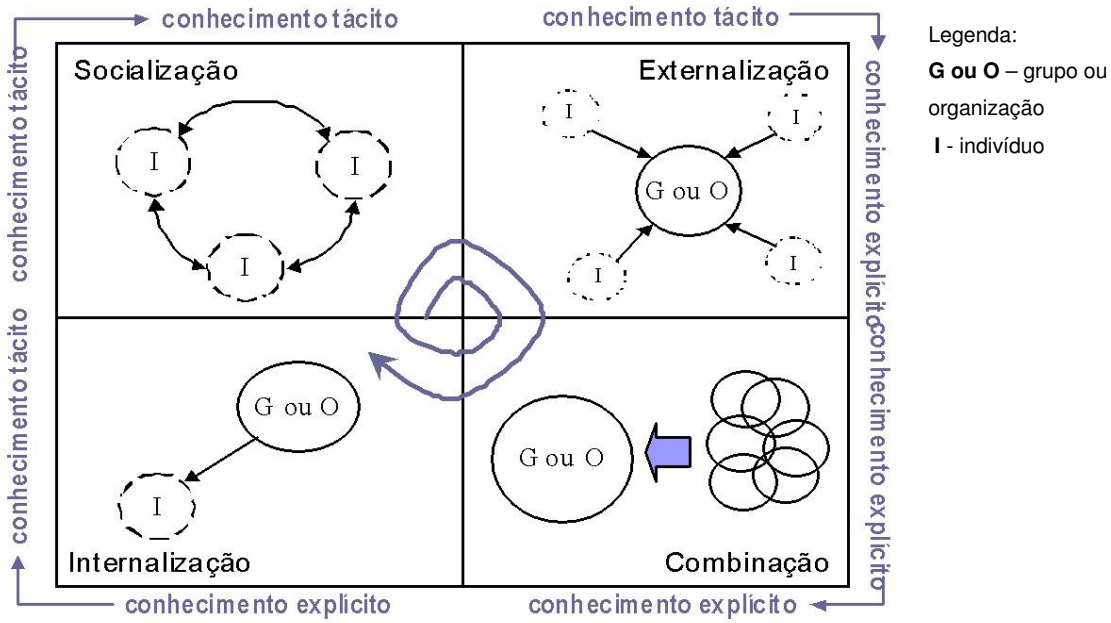


Figura 9: Espiral do conhecimento de Nonaka e Takeuchi (1997) apud Passos (2004)

Os dois tipos de saber são importantes para a organização. Mas, é quando transforma o conhecimento tácito em conhecimento explícito que a organização viabiliza a sua disseminação para um maior número de pessoas e o incorpora ao seu patrimônio, preservando o conhecimento estratégico aos negócios da empresa. A GC está relacionada à capacidade das organizações de utilizarem e combinarem os vários tipos de conhecimento organizacional para desenvolver competências específicas e capacidade inovadora, que se traduzem em novos produtos, serviços e lideranças de mercado. Ou seja, é uma estratégia que viabiliza a criação, o compartilhamento, a reutilização (O'LEARY, 1998c) e o aumento do conhecimento para melhorar o desempenho de uma organização, aumentando sua competitividade e contribuindo para sua sobrevivência (RUS et al., 2001). Villela (2004) distingue muitos dos serviços da GC: descoberta de conhecimento; filtragem de conhecimento; busca de conhecimento; distribuição de conhecimento; associação de conhecimento; e conexão entre pessoas.

O'Leary (1998b) defende que a implementação desses serviços de GC seja apoiada por ferramentas para, por exemplo: autoria, indexação, classificação, armazenamento e recuperação de informação, bem como para compartilhamento e aplicação do conhecimento. Várias tecnologias podem ser utilizadas para fornecer os serviços de GC: correio eletrônico, banco de dados e *data warehouses*, navegadores, máquinas de busca, intranet, bases de conhecimento, modelos de visualização e Internet, sistemas baseados em conhecimento (*knowledge-based systems*), *groupware*, *workflow*, gerência de documentos e agentes inteligentes. Alguns trabalhos citam essas tecnologias: O'Leary (1998a), O'Leary (1998c), Marwick (2001), Rus et al. (2001), Malheiros e Pinho (2003), Mack et al. (2001), Malheiros (2003), Markkula (1999). Aplicações, como gestão de competências, base de soluções e lições aprendidas podem se beneficiar desses serviços (LINDVALL et al., 2002).

Os conceitos e ferramentas de apoio à Gestão de Conhecimento podem ser aplicados a qualquer tipo de organização, inclusive a organizações de software.

Earl (2001) propôs um arcabouço para guiar executivos na seleção/priorização de projetos de GC de acordo com os objetivos, características organizacionais e direcionamentos tecnológicos, comportamentais e econômicos. De acordo com Earl, esforços relacionados à GC podem ser classificados em três grandes categorias: Tecnocrática, Econômica e Comportamental. Cada uma dessas categorias é subdividida em "escolas" (Tabela 1). As "escolas" não são mutuamente exclusivas, mas cada uma delas representa uma orientação em particular.

As escolas categorizadas como Tecnocráticas são baseadas em informação ou na gestão de tecnologias. A filosofia da escola *Sistema* é a codificação da experiência em bases de dados

para que outras pessoas possam consultar. Essas bases de conhecimento podem apoiar a execução de tarefas específicas e se tornam o conhecimento "oficial". A escola *Cartográfica* tem o objetivo de mapear e registrar quem sabe o quê na organização. Como nem todo o conhecimento pode ser explicitado; uma alternativa é identificar quais são as pessoas que podem ser uma fonte de conhecimento em uma organização. O termo "cartográfica" vem do foco em mapear o conhecimento organizacional. A filosofia é conectividade. A idéia por trás da escola *Engenharia* é dar aos trabalhadores a informação que eles precisam para fazer o trabalho. Seu objetivo é apoiar o fluxo de conhecimento, por meio de um processo que deve ser seguido.

Tabela 1: Taxonomia de estratégias para GC. Adaptado de Earl (2001)

	Tecnocrática			Econômica	Comportamental		
Escola	Sistema	Cartográfica	Engenharia	Comercial	Organizacional	Espacial	Estratégica
Foco	Tecnologia	Mapas	Processo	Lucro	Redes	Espaço	*Mindset*
Objetivo	Bases de conhecimento	Diretórios do conhecimento	Fluxo do conhecimento	Ativos de conhecimento	Centros de Conhecimento	Troca de conhecimento	Capacidade de conhecimento
Filosofia	Codificação	Conectividade	Capacidade	Comercialização	Colaboração	Conectividade	Consciência

A escola *Comercial* da categoria "Econômica" está relacionada a tratar o conhecimento organizacional como um ativo para produzir lucro. Palavras como patentes, marcas registradas, e licenças estão ligadas a essa escola. A filosofia é comercializar a propriedade intelectual.

As escolas da categoria "Comportamental" estão focadas em estimular gestores a serem proativos na criação, compartilhamento e uso do conhecimento como um recurso. A escola *Organizacional* está baseada em colaboração. Seu foco é construir redes e comunidades de conhecimento. Uma comunidade é um grupo de pessoas com interesses em comum que trocam e compartilham conhecimentos interativamente. Também baseada em colaboração, mas focada em distribuição espacial das pessoas, a escola *Espacial* estimula a socialização das pessoas como um meio para melhorar a troca de conhecimento. Exemplos de tentativas de aumentar a socialização relacionada à escola *Espacial* são: disposição de máquinas de café em lugares estratégicos ou plantas de escritórios abertas. Por fim, a escola *Estratégica* engloba as outras escolas e sugere que o conhecimento seja tratado como uma mistura de bases de conhecimento, mapas, ferramentas, comunidades, etc. O conhecimento passa a ser fundamental para a organização.

2.4.2 Uso de práticas de GC no desenvolvimento de software

Engenharia de software é uma atividade que lida de forma intensa com o conhecimento (BIRK et al., 1999). Representações e processamentos mentais de conhecimento são atividades complexas requeridas na busca de soluções para os problemas que fazem parte do processo de transformação de uma solicitação abstrata do cliente em declarações de linguagem de programação (VILLELA, 2004). A GC dá bases para o tratamento sistematizado de necessidades

típicas do desenvolvimento/manutenção de software, entre elas (RUS et al., 2001): (i) capturar e compartilhar o conhecimento sobre o processo e sobre o produto; (ii) conhecer o domínio para o qual o software está sendo desenvolvido; (iii) adquirir conhecimento sobre novas tecnologias; (iv) compartilhar conhecimento sobre as políticas locais; (v) saber quem sabe o quê; e (vi) viabilizar colaboração à distância. Com recursos da GC é possível, também, criar uma linguagem comum para o desenvolvimento.

Vários trabalhos relacionam o tema desenvolvimento de software com GC (ex.: Basili et al., 1994b; Oliveira, 1999; Desouza, 2003; Rising, 2004; Ward e Aurum, 2004; Lindvall e Rus, 2003; Santos et al., 2007). Por exemplo, uma rápida pesquisa apenas na ACM Digital Library[14] retorna mais de 730 documentos (utilizando as palavras-chave "*software engineering*" e "*knowledge management*", sem nenhum sinônimo). Mas, mesmo antes do termo "*knowledge management*" se tornar comum nas pesquisas de engenharia de software, a idéia de gerir o conhecimento estava presente, por exemplo, em Basili et al. (1994b), onde a Fábrica de Experiências foi proposta como uma forma de transferir conhecimento. Na literatura disponível, existem *surveys* e relatórios do estado da arte (RUS et al., 2001; CONRADI e FUGGETTA, 2002; LINDVALL e RUS, 2002) e teses de doutorado (BIRK, 2000; FALBO, 1998, BJORSON, 2007) relacionadas. Desde 1989 e 2001, respectivamente, uma conferência internacional em Engenharia de Software e Engenharia do Conhecimento (SEKE) e a Jornada Ibero-americana de *Ingeniería de Software e Ingeniería del Conocimiento* (JIISIC) acontecem, ambas publicando trabalhos sobre GC no desenvolvimento de software. Adicionalmente, Bjornson (2007) conduziu uma revisão sistemática sobre o tema e aponta que algumas escolas de GC ainda não são intensivamente exploradas na engenharia de software. Ele também conclui que mais observações empíricas são importantes para conhecer melhor os efeitos da GC em casos reais da indústria.

É importante ressaltar o Software Engineering Body of Knowledge - SWEBOK (ABRAN e MOORE, 2001), uma iniciativa da IEEE Computer Society e da ACM para montar um guia para o conhecimento existente na literatura a respeito da Engenharia de Software. O SWEBOK pode ser visto como o conteúdo de uma base de conhecimento potencial que poderia ser utilizada por desenvolvedores para resolver diferentes problemas (RUS et al., 2001). Outra iniciativa é a construção de bases de dados para a engenharia de software empírica, como o trabalho do CeBase (*Center for Empirically-Based Software Engineering*). O *FIRE - Framework for Improving the Replication of Experiments* foca na gestão do conhecimento aplicada às replicações de experimentos de engenharia de software (MENDONÇA et al., 2008).

[14] http://portal.acm.org/dl.cfm

O reúso é um dos principais fatores que podem levar ao aumento da qualidade e produtividade do desenvolvimento de software (CARNEIRO, 2003). Pode-se aproveitar: modelos, especificações, código, materiais de treinamento, melhores práticas em engenharia de software e gestão de projetos, processos e lições aprendidas, esta última talvez a mais importante fonte de conhecimento, segundo Markkula (1999). Os serviços de busca de conhecimento, distribuição de conhecimento, associação de conhecimento e conexões entre pessoas podem influenciar positivamente na reutilização de informações. Em função da relevância do tema, boa parte dos trabalhos de GC e o desenvolvimento de software enfocam a reutilização de experiências, componentes ou processos. Alguns trabalhos foram selecionados para ilustrar as diferentes maneiras de reúso que têm sido investigadas e estão resumidos na Tabela 2.

No entanto, segundo Fiorini (2001), a maior parte das pesquisas sobre reúso tem ênfase em código e *design* e poucas pesquisas são sobre reúso de processos. Em complemento, Henninger (2000) afirma que a maioria das pesquisas em desenvolvimento de software enfoca técnicas isoladas, normalmente envolvendo uma nova linguagem, um modelo formal ou uma notação específica. Pouca atenção é dada à compreensão do escopo e limitações dessas soluções pontuais, causando confusão na detecção de quais técnicas devem ser utilizadas e quando. A combinação de soluções pontuais (seja uma nova linguagem ou um modelo formal) para gerar um novo conhecimento, potencialmente mais relevante, pode se beneficiar dos serviços de GC.

Além do reúso, também podem ser úteis para ambientes de desenvolvimento de software: (i) gestão de competências; (ii) capacitação e aprendizado da equipe; (iii) ambiente de colaboração; (iv) formas de distribuição do conhecimento; (v) estabelecimento de linguagens comuns (ontologias); e (vi) portais. Essas questões, apesar de não serem tão exploradas quanto à questão de reúso, também têm sido estudadas no contexto da Engenharia de Software.

Markkula (1999) lista alguns tipos de conhecimento úteis para organizações que desenvolvem e mantêm software: *conhecimento externo* - ponteiros para *sites* da Internet (de clientes, fornecedores e técnicos), manuais, livros, etc.; *conhecimento interno estruturado* - modelos de documentos, relatórios de pesquisa, componentes de software, diretrizes, métodos de planejamento, competência dos funcionários, entre outros; e *conhecimento informal* - discussões, notícias relacionadas a assuntos técnicos, pastas de projeto (sendo que esse último pode ser enquadrado de muito informal até formal). Rus et al. (2001) utilizam uma classificação diferente. Segundo eles, no desenvolvimento de software, podem ser identificados dois tipos de conhecimento: (i) o conhecimento embutido nos produtos gerados (final e intermediários), ambos resultado de atividades intelectuais e criativas e (ii) o "meta-conhecimento" (*meta-knowledge*), que é o conhecimento sobre os produtos e processos.

Tabela 2: Trabalhos sobre reúso de conhecimento no desenvolvimento de software

Foco: Reúso por meio de lições aprendidas
Basili et al. (1994b) propõem a abordagem de uma "fábrica de software" fundamentada em uma base de experiências, que armazena objetos reutilizáveis. O aprendizado ocorre a partir do *feedback* sobre as práticas utilizadas em um projeto para o reúso nas próximas experiências. Henninger (2000) destaca como abordagens baseadas em experiências para desenvolvimento de software podem ser utilizadas para melhorar as práticas de desenvolvimento de software. Borges e Falbo (2001) discutem a importância de armazenar e compartilhar a experiência obtida no desenvolvimento de software para a melhoria contínua da qualidade e produtividade.
Foco: Reúso por meio de uso de padrões, inclusive de padrões de processos
Gamma et al. (1995) sugerem que padrões de projeto (*design patterns*) são uma forma interessante de reúso para software orientado a objeto. Vários trabalhos propondo e aplicando padrões de projeto foram publicados. A disseminação de padrões de projeto e a convicção de que eles facilitam o reúso motivaram o uso de padrões também para processos de desenvolvimento de software. Um exemplo é o trabalho de Harjumaa et al. (2004), que enuncia um conjunto de padrões para direcionar a melhoria de processo de inspeção de software. Já o trabalho de Appleton (1997) apresenta padrões para apoiar a condução de iniciativas de MPS.
Foco: Reúso por meio de *frameworks*
Fayad e Schmidt (1997) propõem *frameworks* de aplicações orientadas a objetos. Um *framework* é uma aplicação semi-completa contendo componentes estáticos e dinâmicos que podem ser adaptados (e reusados) para produzir aplicações específicas. Braga et al. (1999) apresentam o ambiente Odyssey, focado em reúso, de desenvolvimento de software baseado em componentes, para determinados domínios. Carneiro (2003) propõe abordagem iterativa e incremental para o desenvolvimento de *frameworks* orientados a objetos. Dadas as vantagens de utilização de *frameworks*, começaram a surgir trabalhos relacionados a descrever o processo de projetar *frameworks*. Exemplos disso são: um processo para construção e instanciação de *frameworks* baseados em uma linguagem de padrões para um domínio específico (BRAGA e MASIERO, 2002); e a utilização de padrões de projeto para desenvolver *frameworks* orientados a objetos (ROBERTS e JOHNSON, 2005). Cagnin et al. (2004) abordam reúso especificamente na atividade de teste para reduzir custo e esforço de VV&T (Validação, Verificação e Teste) no desenvolvimento de software; Cagnin (2005) apresenta uma contribuição para a reengenharia de software baseada em linguagens de padrões e *frameworks*.
Foco: Reúso por meio de processos, incluindo adaptação e instanciação
Henninger (1999) descreve um ambiente para reutilização de processos. A proposta utiliza um sistema baseado em regras para adaptar o processo de software para as necessidades de projetos individuais. Também investigando adaptação de processos, Fiorini (2001) propõe uma arquitetura para organização e descrição de processos, visando à reutilização de processos. Oliveira (1999) explora a reutilização de processos por meio de especialização e instanciação de processos. Ahonen et al. (2002) analisam o processo de reúso utilizado em quatro estudos de caso para a melhoria desse processo de reúso.
Foco: Reúso por meio de garantia da qualidade
Boegh et al. (1999) propõem o SQUID, um método e uma ferramenta para garantia da qualidade de software, que permite que uma organização de desenvolvimento de software planeje e controle a qualidade do produto durante o desenvolvimento.
Foco: Reúso por meio de linha de produto
Clements e Northrop (2002) e Northrop (2006) descrevem uma estratégia de reutilização nomeada linha de produto. Uma linha de produto de software é um conjunto de sistemas que são desenvolvidos a partir de um conjunto comum de ativos principais ao invés de projeto a projeto.
Foco: Reúso por meio de ambientes de desenvolvimento
Falbo et al. (2003) descrevem um ambiente de desenvolvimento de software, centrado em processos e fundamentado em ontologias. Uma das características do ambiente é a presença de uma base de conhecimento que permite que o ambiente ofereça um suporte especializado ao usuário na realização de suas tarefas e possibilita que as informações geradas mantenham-se interligadas e consistentes ao longo de todo o processo. Ontologias podem ser definidas como especificações conceituais de um domínio de problema (O'LEARY, 1998b), isto é, uma descrição de conceitos e relações que podem existir nesse domínio. Uma ontologia é constituída por um vocabulário específico usado para descrever uma realidade e por um conjunto de suposições referentes ao significado pretendido para os termos usados no vocabulário. O uso de ontologias pode facilitar a explicitação e posterior internalização do conhecimento porque é uma forma de estabelecer um acordo sobre o conhecimento a ser trocado entre os seus usuários.
Foco: Reúso por meio de pacotes de experimentação
Shull et al. (2004) argumentam que a disponibilidade de pacotes de laboratório para experimentos pode incentivar melhores replicações de experimentos e estudos complementares e descreve uma instanciação do modelo de Nonaka e Takeuchi (1997) para experimentação em engenharia de software. Mendonça et al. (2008) derivam um *framework* para coordenar e evoluir experimentos controlados de engenharia de software, a partir do conhecimento adquirido e das experiências compartilhadas nas replicações de um experimento.

O maior desafio para gerenciar o conhecimento é que boa parte do conhecimento em engenharia de software é tácito (no sentido de não documentado) e, dificilmente, é dedicado tempo a torná-lo explícito. Questões relacionadas a maior ou menor explicitação (documentação) do conhecimento precisam ser consideradas. Tais questões estão sendo abordadas também nas discussões sobre balanceamento entre metodologias mais ágeis ou mais planejadas (Seção 2.2.2).

Uma das formas de explicitar o conhecimento sobre o desenvolvimento de software é documentar um processo de desenvolvimento em artefatos, procedimentos, métodos e modelos.

Iniciativas no sentido de intensificar a documentação do processo de desenvolvimento de software e dos projetos de software foram propostas por vários autores. Boa parte deles defende a adoção de um processo padrão que possa ser adaptado a projetos específicos, de acordo com suas características (CAMPOS et al., 2006; FIORINI, 2001; SEI, 2006; SOFTEX, 2009). Essa abordagem converge para os princípios da GC, no que tange a reutilização, de duas formas:

- a adoção de processos padrão permite que, com pequenas adaptações, o "como fazer" esteja explicitado de forma que, facilmente, é possível identificar a forma de trabalho de uma organização; e
- esses processos normalmente definem uma forma padrão de armazenamento de informações em ferramentas ou artefatos. São recomendados artefatos padrão para os projetos. Dessa forma, é mais fácil identificar informações de projetos similares para reutilização em novos projetos. Essa reutilização pode acontecer tanto em relação ao produto final como em relação aos artefatos intermediários do projeto.

Nesse ponto, vale ressaltar que a automatização desse processo (Seção 2.2.1) pode ser muito interessante. Uma ferramenta pode orientar a execução correta das atividades e viabilizar o armazenamento do conhecimento acerca da garantia de qualidade de forma estruturada.

O estabelecimento de ontologias pode contribuir para aquisição, organização, compartilhamento e reutilização de conhecimento de engenharia de software. O uso de ontologias pode contribuir também para a definição de uma base de conhecimentos. Exemplos na área de engenharia de software são as ontologias para: processos de software (FALBO, 1998); requisitos (LIN et al., 1996); teste de software (HUO et al., 2003; BARBOSA et al., 2006); domínios (OLIVEIRA, 1999); documentação (NUNES et al., 2004); riscos (FALBO et al., 2004); qualidade de software (DUARTE e FALBO, 2000). Também foram propostas/desenvolvidas ferramentas de apoio à construção de ontologias, como: Protégé (PROTÉGÉ, 2007) e ambientes de desenvolvimento de software baseados em ontologias (FALBO et al., 2003).

2.4.3 Ambientes e ferramentas de GC

A tecnologia tem um papel relevante na gestão direta do saber e é essencial para o reúso do conhecimento. Ela facilita a gestão do conhecimento explícito, fornecendo a estrutura para retenção, compartilhamento e distribuição de conteúdos por diversos meios, como: correio eletrônico, portais, intranet, sistemas de fluxo de trabalho, ambientes de bate-papo e mensagens instantâneas e banco de dados. Esta seção traz um panorama dos recursos tecnológicos que vêm sendo aplicados para autoria; indexação; classificação; armazenamento; recuperação de informações ou trabalho colaborativo, com ênfase na sua aplicação para o desenvolvimento de software. O modelo de arquitetura de GC apresentado em Lindvall et al. (2002) serve como baliza para descrever esses recursos tecnológicos (Figura 10).

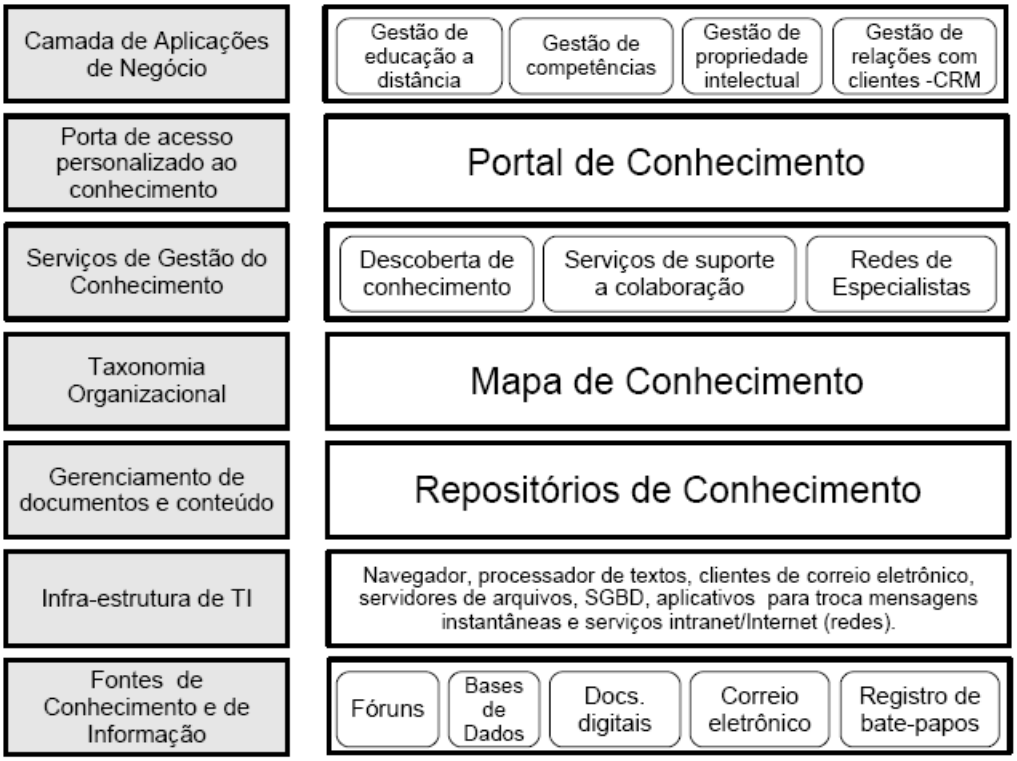


Figura 10: Modelo de arquitetura de GC. Adaptado de Lindvall et al. (2002)

A camada mais baixa representa repositórios de documentos como correio eletrônico, registros em bancos de dados e históricos de bate-papos. Essa camada é composta por fontes de informação de conhecimento explícito, que é capturado por meio da segunda camada de infraestrutura tecnológica (editores de texto, sistemas de gerenciamento de bancos de dados (SGBD), programas de correio eletrônico, etc.). A terceira camada, de gerenciamento de documentos e de conteúdo, com funcionalidades de busca, recuperação e análise de informações, é representada pelos repositórios de conhecimento. Ferramentas de classificação e indexação

representam a camada de taxonomia organizacional, que tem o objetivo de organizar o conhecimento em mapas de conhecimento (quarta camada).

A quinta camada aglutina os serviços de GC (ex: busca, distribuição, associação) e é materializada como portais, serviços de suporte à colaboração, ferramentas de mineração de dados ou serviços de registro da memória organizacional (redes de especialistas, base de soluções e de lições aprendidas).

Esses serviços suportam aplicações de suporte ao negócio, como gestão de competências, gestão do relacionamento com o cliente, entre outras (sétima camada), que podem ser acessadas por meio de um portal de conhecimento (sexta camada).

Como já foi dito, a Engenharia de Software lida de forma intensa com o conhecimento e, por isso, pode se beneficiar de cada camada do modelo arquitetural apresentado. Pontos de interseção mais específicos entre o desenvolvimento de software e algumas camadas apresentadas (Figura 10) são discutidos a seguir.

Repositórios de Conhecimento – Camada Gerenciamento de documentos e conteúdo

O gerenciamento de documentos e conteúdo permite que documentos sejam compartilhados e acessados, inclusive remotamente. O uso de sistemas de gestão de conteúdo e documentos permite o armazenamento, classificação, e controle de versão de documentos. Alguns sistemas de gerenciamento de documentos e conteúdo também permitem identificação de especialistas a partir do registro da autoria do documento. Um exemplo comercial é o Hyperwave[15]. Soluções para a Internet, em geral desenvolvidas e distribuídas como software livre, têm se tornado popular. Um bom exemplo é o Plone[16]. Ele facilita a organização, o controle e a publicação de documentos e arquivos multimídia. Os documentos gerenciados pelo Plone ficam disponíveis por meio de páginas web dinâmicas. Isto permite que o Plone possa ser utilizado como ferramenta de apoio para a construção e manutenção de portais, além dos benefícios diretos de acessibilidade e manutenção advindos do uso da plataforma web. O projeto Plonetarium, por exemplo, é uma especialização do Plone, especificamente voltada para a criação de portais e publicação de livros, revistas, enciclopédias, material didático, bancos de dados, entre outros. O Plonetarium é desenvolvido no âmbito do projeto TIDIA[17] e

[15] http://www.hyperwave.com/e/technologies/document_management
[16] http://plone.org/
[17] http://www.tidia.fapesp.br/portal

disponibilizado como uma das ferramentas da Incubadora Virtual da FAPESP (FAPESP - Fundação de Amparo a Pesquisa do Estado de São Paulo, 2006).

Outra especialização, bastante difundida, de sistemas de gerenciamento de conteúdo é o software de manutenção de wikis. O modelo Wiki é uma rede de páginas web contendo as mais diversas informações, que podem ser modificadas e ampliadas por meio de navegadores comuns (WIKIPEDIA, 2010a). Wiki é um software colaborativo que permite a edição coletiva dos documentos e é possível acrescentar e editar suas páginas diretamente, e não apenas documentos armazenados. Também é comum que seja permitida a edição por usuários anônimos. A Wikipédia[18] segue este estilo de operação. Exemplos específicos para a engenharia de software de sistemas de gerenciamento de documentos e conteúdos são os sistemas de gestão de requisitos e os sistemas de gestão de configuração.

O gerenciamento de requisitos é uma atividade da engenharia de software estreitamente ligada à gestão de conhecimentos. Particularmente, a rastreabilidade garante acesso à informação de qual requisito originou determinada parte do código e também dos modelos de análise e de projeto. Essa rastreabilidade apóia a análise de impacto gerado por mudanças nos requisitos e apóia atividades de verificação e validação. Um exemplo de ferramentas de gerenciamento de requisitos é a Rational Requisite Pro[19].

Todas as fases do desenvolvimento de software geram artefatos. O próprio código pode conceitualmente ser visto como um conjunto de documentos. Assim, sistemas de gerenciamento de documentos e conteúdo podem contribuir significativamente para o desenvolvimento de software. As ferramentas de controle de configuração são exemplos disso. Essas ferramentas mantêm a integridade dos artefatos e dos produtos gerados durante o ciclo de vida do software. Além disso, permitem o rastreamento de qual versão de cada parte do código corresponde ao produto que está em produção. Exemplos de itens que são mantidos sob gestão de configuração: planos de projeto, cronogramas, especificações de caso de uso, modelos ou partes de um modelo, programas-fonte, módulos executáveis ou componentes, arquivos de ajuda, roteiros de teste. Para manipular esses itens, é comum que essas ferramentas ofereçam funcionalidades de: carga e armazenamento de documentos; controle de versão; geração de linhas de referência (*baselines*); acesso remoto (muitas vezes via Internet) e compartilhado aos itens de configuração e; controle de concorrência. Existem várias ferramentas de gestão de configuração, como: Visual Source

[18] http://pt.wikipedia.org
[19] http://www-306.ibm.com/software/awdtools/reqpro/

Safe[20], Rational Clear Case[21], PVCS Professional Suite[22]. Opções de software livre para gestão de configuração são, por exemplo, *Concurrent Versions System* – CVS ou Subversion-SVN [23].

Os sistemas de gestão de configuração podem ser utilizados para criar, indiretamente, serviços de memória organizacional (quem fez o quê, quando e por que). Tais sistemas geram/ armazenam meta-conhecimento sobre os produtos e artefatos (ex.: quando foi armazenado, quem armazenou e histórico de operações). Além disso, muitos desses sistemas permitem comparar documentos e identificar o que foi alterado em uma nova versão. Sistemas de gestão de configuração podem apoiar a documentação de processos de desenvolvimento de software.

Portal de Conhecimento – Camada Portal de acesso personalizado ao conhecimento

O portal de conhecimento (Figura 10, sexta camada) cria um acesso único e personalizado para uma grande e heterogênea coleção de dados, informações e conhecimento (LINDVALL et al., 2002). A utilização de tecnologia web (HTML[24] e XML[25]) para o desenvolvimento de portais é amplamente difundida.

O acesso único deve ser interpretado como uma porta de acesso personalizada para uma comunidade de usuários que compartilhem interesses comuns. A abrangência de informações e conhecimento disponíveis, a partir de um portal, é variável. Ela tanto pode ser centrada em torno de um conteúdo específico, como pode abranger assuntos e serviços diversos, mas de interesse de uma comunidade específica. O Portal Java[26] é um exemplo de portal de conhecimento para usuários que tenham interesse especificamente em Java. Este também seria o caso de um portal sobre engenharia de software, ou ainda sobre a MPS, como, por exemplo, o portal do MR-MPS.

O acesso aos repositórios de conhecimento e às aplicações de gestão do conhecimento para apoio ao negócio pode ser feito por meio de portais.

---

[20] http://msdn2.microsoft.com/en-us/vstudio/aa700900.aspx
[21] http://www-306.ibm.com/software/awdtools/clearcase/
[22] http://www.serena.com/US/products/pvcs/index.aspx
[23] CVS (http://www.nongnu.org/cvs/) e Subversion (http://subversion.apache.org) são sistemas de controle de versão de código aberto, muito usado pelas comunidades de software livre.
[24] HTML - *Hyper Text Markup Language*, linguagem de marcação de hipertexto utilizada para construir páginas web, visualizadas por meio de programas chamados navegadores (ou *browsers*).
[25] *eXtensible Markup Language*, linguagem baseada em texto, criada pelo consórcio W3C, que está se tornando um padrão para troca de informações entre sistemas (BRAY, 2008).
[26] http://www.portaljava.com/home/index.php

Gestão de Competências – Camada de aplicações de negócio

Dentre as aplicações de apoio ao negócio (Figura 10, sétima camada), as aplicações de gestão de competências mantêm um registro das habilidades dos empregados de uma organização. Conhecendo o perfil e as competências individuais de cada empregado, a empresa é capaz de localizar profissionais que possuam competência em determinado assunto e qual o nível de seu conhecimento (conhece, domina ou aplica, por exemplo). Nem todo conhecimento organizacional pode ser explicitado. Parte dele permanece na cabeça dos especialistas da organização. Em pequenas organizações as pessoas normalmente sabem quem têm o domínio de determinado conhecimento. Mas, à medida que aumenta o número de funcionários de uma empresa, aumenta a dificuldade de saber "quem sabe o quê". Segundo Rus et al. (2001), os sistemas de gestão de competências, além de facilitar a identificação de funcionários para alocação em projetos, podem ser utilizados para *marketing* externo ou como insumo para promover o desenvolvimento de pessoal. Uma das questões mais complexas da gestão de competências é identificar quem realmente domina um assunto. Uma maneira de fazê-lo seria o preenchimento de perfil pelo próprio especialista. Outra maneira poderia ser o mapeamento de especialistas a partir dos documentos de sua autoria (RUS et al., 2001). A plataforma Lattes[27] é um exemplo de aplicação de gestão de competências. Estudos na área de visualização de informação também podem contribuir para a melhoria dos sistemas de gestão de competências. A solução proposta por Passos (2004), CompMAP, utiliza-se desse recurso.

Educação à distância – Camada de aplicações de negócio

A aquisição de conhecimento tácito é pré-requisito para realizar uma nova atividade (MARWICK, 2001). Para utilizar determinada informação, os indivíduos precisam entendê-la e internalizá-la. A leitura de documentos permite a absorção de experiências de outras pessoas e a combinação do conhecimento tácito de cada indivíduo com o conhecimento explicitado por outros. A tecnologia pode auxiliar os usuários a transformar conhecimento explícito em conhecimento tácito, não só por viabilizar o armazenamento e a recuperação de informações, como também por facilitar o aprendizado. Um caso especial é a educação à distância. Essa modalidade de ensino auxilia a disseminação do conhecimento organizacional, em particular em organizações que trabalham com DDS, uma vez que os cursos e respectivos conteúdos podem ser utilizados remotamente e por todas as pessoas que deles necessitem para executar suas

[27] http://lattes.cnpq.br/

atividades. Além disto, facilita a participação dos empregados, na medida em que permite a programação personalizada dos períodos dedicados ao ensino/aprendizagem (SERPRO, 2000).

Em Barbosa (2004), são discutidos e estabelecidos mecanismos de apoio à atividade de modelagem de conteúdos e ao processo de desenvolvimento de módulos educacionais, culminando com a proposta de um modelo de maturidade de processos - CMM/Educacional, que visa a disponibilizar material didático e de treinamento livre, na mesma concepção que produtos de software livre.

Sistemas de fluxo de trabalho (*workflow*) visam a garantir e acelerar o fluxo de processos pré-programados, dando suporte ao trabalho cooperativo. Para isto, esses sistemas acompanham os passos e a execução de cada uma das atividades destes processos. Geralmente, esses sistemas definem papéis para os seus usuários, rotas para as informações a serem processadas, e regras de como estas informações devem ser processadas. Sistemas de fluxo de trabalho tornam explícito o conhecimento associado aos processos de negócio de uma organização.

Os sistemas de gerência de fluxo de trabalho (*workflow*) fornecem ferramentas de software de apoio para a definição e execução de fluxos de trabalho. Esses sistemas, como Lotus Notes[28], Rational Clear Quest[29] ou ARS Remedy[30], podem ser adaptados para uso no domínio de desenvolvimento de software.

No desenvolvimento de um software, vários processos podem ser mapeados e apoiados por sistemas de fluxo de trabalho, inclusive todo o ciclo de vida do desenvolvimento. Recentemente, vários trabalhos têm sido direcionados para o estudo de ambiente de engenharia de software orientado a processos (Seção 2.2.1).

Especificamente no domínio de testes, pode-se dizer que as ferramentas de rastreamento de erros (*bug tracking*) têm características dos sistemas de fluxo de trabalho para atividades específicas do teste de software. Elas são aplicações de software que dão apoio a gestão do tratamento de erros encontrados em um software. Bugzilla[31] e Mantis Bug Tracker[32] são exemplos dessas ferramentas. A mesma filosofia das ferramentas de rastreamento de erros foi utilizada para o tratamento de propostas de melhorias de processo.

[28] http://www-306.ibm.com/software/info/ecatalog/pt BR/products/I105893X30130U84.html
[29] http://www-306.ibm.com/software/awdtools/clearquest/index.html
[30] http://www.bmc.com/products/proddocview/0,2832,19052 19429 22735072 106757,00.html
[31] https://bugzilla.mozilla.org/
[32] http://www.mantisbt.org/

2.5 Desenvolvimento distribuído e desenvolvimento de software livre

A distribuição de desenvolvedores no tempo e no espaço vem se tornando prática comum nas organizações e muitas delas começaram a investir em ambientes de desenvolvimento distribuído de software (DDS) e *outsourcing*. O estudo envolvendo o DDS transcende a Ciência da Computação, sendo uma área multidisciplinar envolvendo também Sociologia, Psicologia, Administração e Educação (CARMEL, 1999 apud PRIKLADNICKI, 2003). Diferentes perspectivas do desenvolvimento distribuído têm sido exploradas. Ambientes e ferramentas têm sido criados para ajudar no controle e coordenação das equipes distribuídas. O DDS tem sido caracterizado pela colaboração e cooperação entre unidades de organizações e pela criação de grupos de trabalhos que desenvolvem um projeto, mas estão localizados em cidades ou países diferentes (KIEL, 2003). O foco da investigação de DDS neste trabalho foi em questões associadas à colaboração e a cooperação entre equipes distribuídas.

Um caso particular de desenvolvimento distribuído é o desenvolvimento de software livre. O movimento de SL vem ganhando espaço no mercado e despertando o interesse da academia, do governo e da indústria. Software livre é qualquer software cuja licença garanta ao seu usuário certas liberdades relacionadas ao uso, alteração e redistribuição (FREE SOFTWARE FOUNDATION, 2009). Seu aspecto fundamental é o fato do código-fonte estar livremente disponível para ser lido, estudado ou modificado por qualquer pessoa interessada (REIS, 2003b).

Ambos, DDS e o desenvolvimento de SL serão abordados nesta seção.

2.5.1 Desenvolvimento distribuído de software (DDS)

O DDS está exposto a vários desafios, entre eles, problemas de comunicação e coordenação de equipes e das atividades; e controle de artefatos (MAIDANTCHIK, 1999; PRIKLADNICKI et al., 2004). Os aspectos mais críticos no DDS são (MAIDANTCHIK, 1999): a troca de informações; o acesso ao conhecimento; a realização de atividades cooperativas; a coordenação de tarefas assíncronas e o registro de atividades cooperativas para posterior reutilização. Tais problemas podem ser reduzidos com o estabelecimento e adoção de um processo bem definido, adequado às especificidades dos ambientes distribuídos (MAIDANTCHICK et al., 1999). Ramasubbu e Balan (2007) complementam que os efeitos do espalhamento podem ser mitigados significativamente com o uso de processos estruturados de

engenharia de software, tornando o processo de desenvolvimento um fator crítico para DDS (PRIKLADNICKI et al., 2004).

Dentro desse contexto, Maidantchick (1999) propôs um processo padrão para DDS, que estende a norma ISO/IEC 12207, introduzindo processos e atividades específicas para atender às necessidades de equipes de desenvolvimento distribuídas, tais como processos de: comunicação, coordenação e controle de artefatos. Os três processos são importantes para uma MPS distribuída e colaborativa e são retomados durante a definição da ColabSPI, já que, similarmente ao DDS, uma MPS distribuída e colaborativa dependerá de comunicação, coordenação de grupos distribuídos e gestão de mudanças e controle de artefatos.

Ampliando a visão de processo de desenvolvimento de software, Prikladinick (2003) sugere um modelo de referência para DDS que trabalha também com a visão organizacional e de projeto. Esse modelo aborda três diferentes níveis de maturidade da organização para trabalhar o DDS. Na Figura 11, que apresenta o nível otimizado desse modelo, é possível observar duas etapas diferenciadas de planejamento: (i) estratégico, que acontece na esfera organizacional e (ii) tático/operacional , que é derivado do planejamento estratégico e acontece na esfera de projetos (no âmbito da unidade de desenvolvimento distribuída). É possível, ainda, observar um ciclo de *feedback* entre as unidades e a matriz.

A estrutura organizacional proposta em ColabSPI (Seção 5.4) trabalha com essa visão de um planejamento estratégico central (corporativo) e o desdobramento desse planejamento nas unidades (local). Um exemplo de modelo de plano de gestão da MPS corporativo está disponível no Apêndice A.

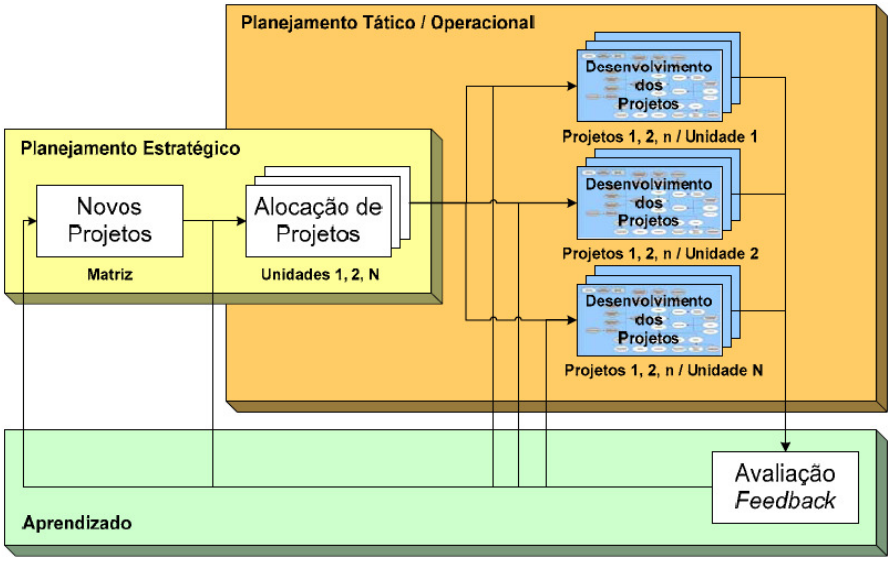


Figura 11: Modelo de referência para o DDS. Extraído de Prikladnicki (2003)

Colaboração

O DDS tem sido caracterizado por colaboração e cooperação entre unidades de organizações e por grupos de desenvolvedores que trabalham em conjunto, mas estão localizados em cidades ou países diferentes, distantes temporal e fisicamente (PRIKLADNICKI, 2003).

A colaboração pode ser definida como um trabalho desenvolvido por duas ou mais pessoas para atingir um objetivo em comum. Um grupo tem maior capacidade de gerar alternativas, levantar suas vantagens e desvantagens, selecionar as viáveis e tomar decisões (GEROSA et al.; 2003).

O uso de sistemas de gestão de conteúdo e documentos permite o armazenamento, classificação, e controle de versão de documentos; permitindo, de certa forma, um trabalho colaborativo. O compartilhamento e evolução de idéias, no entanto, extrapola a simples gestão de documentos e conteúdo. Aspectos psicológicos, sociais e técnicos devem ser estudados em conjunto para entender as características de trabalhos cooperativos. Esse é o foco de estudos sobre Trabalho Cooperativo Suportado por Computador (CSCW, *Computer Supported Cooperative Work*) (GRUDIN, 1994). A conferência internacional *Computer Supported Cooperative Work in Design* (CSCWD) é direcionada para o tema, em particular para como a cooperação de diferentes equipes pode contribuir para projetar artefatos complexos, sendo suas discussões interessantes tanto para o desenvolvimento distribuído de software, quanto para a melhoria colaborativa de processos.

O CSCW pode ser classificado de acordo com duas dimensões: **tempo**, o trabalho acontece de forma síncrona ou assíncrona; e **local** - as pessoas estão no mesmo local ou em locais diferentes (Figura 12).

Pesquisas sobre *groupware* (COLEMAN e KHANNA, 1995) abordam questões relativas à realização de atividades colaborativas, como reuniões, por equipes distribuídas geograficamente. O foco é o apoio computacional à comunicação, ao compartilhamento de informações e recursos, e à coordenação de tarefas (ELLIS et al., 1991). Pesquisas em engenharia de software, algumas vezes classificadas *collaborative software engineering* ou *collaborative software development*, mencionam a importância da colaboração para o ciclo de desenvolvimento (FUGGETTA, 2000; HUPFER et al., 2004; STOREY, et al., 2005; WHITEHEAD, 2007), inclusive para o DDS (GUTWIN et al. 2004). Aplicados a engenharia de software, os conceitos de *groupware* podem contribuir para, por exemplo, identificação de requisitos (SOMMERVILLE et al., 1998; LAPORTI et al., 2009), modelagem e codificação (WERNER et al., 2003), inspeção (BROTHERS et al., 1990; GRUNBACHER et al., 2003) e

também para compreensão e execução do processo de software em um projeto (ARAUJO e BORGES, 2007).

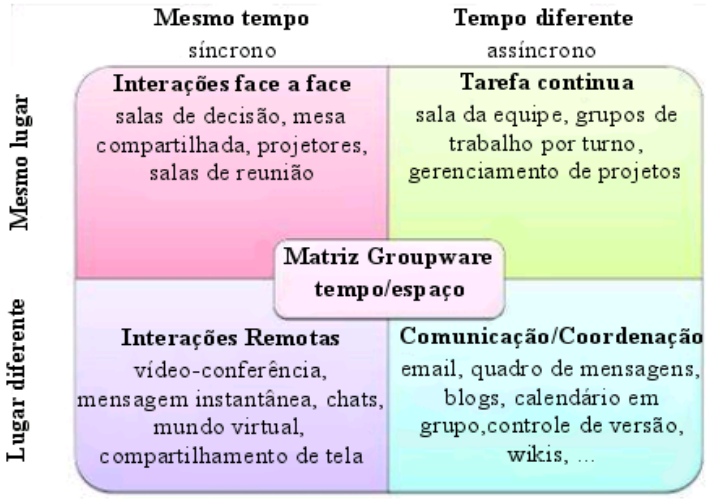


Figura 12: Matriz CSCW de duas dimensões. Adaptado de Wikipedia (2010b)

Ambientes de desenvolvimento colaborativo (ADC) disponibilizam canais de comunicação que garantem o acesso contínuo à informação e contribuem para que as pessoas trabalhem em momentos distintos e em locais diferentes de forma coordenada, por meio da Internet. Os ADC reúnem diversas ferramentas e criam um ambiente para cada projeto, utilizando conceitos de aplicações voltadas para a Web, redes sociais e ferramentas de comunicação. Tipicamente um ADC disponibiliza ferramentas de correio eletrônico, fóruns de discussão, controle de versão de documentos (gestão dos documentos), *wiki*, quadros de aviso, agenda e acompanhamento de atividades. Um ADC pode apoiar o DDS já que, segundo Stein et al. (1997) apud Maidantchik (1999), um ambiente de DDS deve apoiar, entre outras atividades: o compartilhamento de informação; o registro e o compartilhamento de tópicos de discussão; histórico do trabalho e ligação a documentos de apoio.

Booch e Brown (2002) classificam as funcionalidades necessárias para um ADC em três "C": coordenação (ex: controle de versão ou acompanhamento de tarefas), comunicação (fóruns ou navegação compartilhada de documentos) e construção de comunidades (auto-gestão de projetos, estabelecimento de protocolos). De acordo com os autores, há indícios de que muita energia e tempo do desenvolvimento são investidos em atividades não direcionadas para o objetivo final: produzir o software. Exemplos dessas atividades são (BOOCH e BROWN, 2002): (i) iniciação de projetos, incluindo o tempo demandado para achar as ferramentas corretas, descobrir com quem falar e entender estado atual do projeto; (ii) manutenção da comunicação

efetiva entre o grupo, incluindo a distribuição de informações como: situação do projeto; memória do projeto ou a experiência da equipe; (iii) gerenciamento do tempo (e falta de recursos) entre as tarefas; (iv) negociação com partes interessadas; (v) coisas que não funcionam. Partindo da premissa que "processo de software é software também" é possível assumir que tais atividades se aplicarão a evolução do processo também.

Hupfer et al. (2004) apresentam o Jazz[33], ambiente para desenvolvimento colaborativo de aplicações, e uma abordagem para colaboração contextual (*contextual collaboration*), onde os usuários não precisam sair para ambientes colaborativos para se comunicar, pois as capacidades de colaboração são disponíveis como componentes que estendem as aplicações padrão de desenvolvimento.

Ambientes de desenvolvimento integrados (*Integrated Development Environments* - IDEs) de alguma forma já abordam algumas dessas questões (como “i” e “v”) e, muitas vezes, estão integrados com repositórios de configuração.

No entanto, um ADC pode ir além, apoiando a execução das outras atividades (a saber: “i”, “ii”, “iii” e “v”), permitindo o redirecionamento da energia do projeto para o seu objetivo final. Booch e Brown (2002) advogam que o uso de *Collaborative Development Environments* (CDE) pode diminuir atritos no desenvolvimento. Apenas ter um endereço de Internet único pode, por exemplo, reduzir o tempo de iniciação de projetos (todas as informações estão em um mesmo local). O gasto de energia dos desenvolvedores com problemas como integração de ferramentas com diferentes interfaces e pequenas interrupções para resolução de problemas de tecnologia e ferramentas isoladas pode ser menor. A auto-gestão, por sua vez, permite à equipe gerenciar suas atividades e artefatos sem terceiros (menos partes interessadas e menos problemas de comunicação).

Pesquisas em *outsourcing* exploram os benefícios desses tipos de ambiente colaborativos. Contudo, foi no desenvolvimento de software livre (FREE SOFTWARE FOUNDATION, 2008) que tais ambientes tiveram visibilidade. Vários ADC são utilizados pelas comunidades para apoiar o desenvolvimento de software. Um exemplo é o SourceForge[34], ramo público do Collab.net[35]. Segundo seu *site*[34], o ambiente é o maior repositório mundial de software livre (SL). Dado que muitas características do desenvolvimento distribuído em uma organização

[33] http://www-01.ibm.com/software/br/rational/info/features/collaboration/index.html
[34] Informações sobre o SourceForge disponíveis em: http://sourceforge.net/about
[35] Informações sobre o Collabnet disponíveis em: http://www.collab.net/

podem ser relacionadas às características do desenvolvimento de SL, características específicas do desenvolvimento de SL foram investigadas (Seção 2.5.2).

2.5.2 Desenvolvimento de software livre

A denominação software livre (*free software*) designa o software para o qual o usuário possui as liberdades de (FREE SOFTWARE FOUNDATION, 2009):

0. Executar o programa, para qualquer propósito;
1. Estudar como o mesmo funciona e adaptá-lo para as suas necessidades;
2. Redistribuir cópias de modo a ajudar ao próximo; e
3. Aperfeiçoar o programa e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie.

Embora código aberto e software livre tratem de desenvolver programas com códigos abertos, coletivamente, e de conferir liberdade de uso desses programas, pode-se dizer que existem algumas diferenças ideológicas entre os dois movimentos. Do ponto de vista deste trabalho de doutorado, essa diferenciação não é significativa, já que as definições oficiais de ambos são as mesmas e há um conjunto de atividades de desenvolvimento que são organizadas da mesma forma, seja para software livre ou para código aberto. Essas atividades de desenvolvimento têm características do Modelo Bazar: horizontal, coletiva e cooperativa, baseada nas comunidades de software livre (RAYMOND, 1999; SOFTEX, 2005). Isto posto, a denominação software livre será utilizada, daqui para frente, indistintamente para tanto para software livre como para código aberto.

Os princípios do software livre fundamentam-se nas premissas básicas de liberdade de expressão, acesso à informação e do caráter eminentemente coletivo do conhecimento, que deve ser construído e disponibilizado democraticamente, e não privatizado (SOFTEX, 2005). Para que o software seja realmente livre o código-fonte deve acompanhá-lo. Um mecanismo que garante tais liberdades é a licença de software livre. Uma licença de software é o documento que prescreve como software poderá ser utilizado, distribuído e copiado. As licenças de software livre têm a força de um contrato de adesão, no qual o usuário compromete-se a respeitar as regras propostas pelo proprietário dos direitos autorais. A GNU GPL (GNU *General Public License*) (FREE SOFTWARE FOUNDATION, 2008) é a mais utilizada delas.

A idéia de software livre tem sido popularizada a partir de 1985, quando Richard Stallman fundou a FSF (FREE SOFTWARE FOUNDATION, 2009), fundação cujo objetivo é promover o desenvolvimento e a utilização de software livre. É possível observar a intensificação dos movimentos estratégicos em torno do software livre nos mais diversos

segmentos da sociedade. Em nível mundial, o desenvolvimento de software livre está evoluindo, de uma fase inicial amadora, para um cenário de formação de grandes consórcios, que contam com a participação de governos, academia, comunidades de software livre e grandes empresas. Exemplos disso são os consórcios Qualipso (Seção 2.5.3) e o ObjectWeb, surgido da iniciativa do INRIA (*Institut National de Recherche en Informatique et en Automatique*), da France Telecom e da Bull na França e que hoje, assim como o Qualipso, conta com participantes de diversas partes do mundo.

No Brasil, o software livre tem assumido importância crescente em universidades, em empresas privadas e nos órgãos, empresas e autarquias do Governo Federal, Estadual e Municipal. Exemplo disso é a diretriz do governo brasileiro no sentido de aumentar o uso de software livre no âmbito do governo federal. O Observatório Econômico da Softex e o Departamento de Política Científica e Tecnológica da Unicamp realizaram, com o apoio do MCT (Ministério da Ciência e Tecnologia), uma pesquisa para estudar o impacto do software livre no Brasil (SOFTEX, 2005). Foram estudados aspectos como: (i) abrangência de utilização; (ii) capacitação dos desenvolvedores; e, particularmente, (iii) os impactos que dizem respeito às empresas de software (capacitação, modelos de negócio, etc.). O grupo registra que o software livre desponta como opção estratégica para o desenvolvimento tecnológico com vistas à inclusão social, a partir de diversas localidades do Brasil (SOFTEX, 2005).

O software livre tem em sua essência alguns componentes estruturais, entre eles: (i) o uso de padrões abertos; (ii) o licenciamento livre dos softwares; (iii) a formação de comunidades, em especial de usuários e desenvolvedores. O governo brasileiro, como incentivador do software livre, tem tratado dos três temas em paralelo, por meio de ações complementares, destacando-se: (a) a e-ping, padrões de interoperabilidade do governo federal; (b) estudos sobre licenciamento livre do software, encomendado pelo Instituto Nacional de Tecnologia da Informação – ITI; e (c) Comitê Técnico para Implementação do Software Livre – CISL, ponto de encontro da comunidade do governo de software livre[36]. Um das experiências, o Portal do Software Público Brasileiro, é um modelo de atuação do Estado para promoção de redes colaborativas e estruturação de um portal de soluções livres. Este portal pretende facilitar a implantação de novas ferramentas; promover a integração entre as unidades federativas; e oferecer um conjunto de serviços públicos para sociedade com base no bem software[37]. Mais recentemente, foi criado

[36] http://www.softwarelivre.gov.br/comunidade-no-governo
[37] http://www.softwarepublico.gov.br/O_que_e_o_SPB

o projeto Demoiselle[38], cujo objetivo é disponibilizar uma arquitetura de software e um processo de desenvolvimento de software que possam ser livremente utilizados por organizações que desenvolvam software para o governo brasileiro. Um terceiro exemplo de disseminação nacional de um software livre, o Ginga, permite a produção de conteúdos audiovisuais digitais (TV digital interativa) interoperáveis e de acesso gratuito aos interessados.

Com o software livre surgiram novas maneiras de desenvolver software. Características marcantes do desenvolvimento de SL são apresentadas na Tabela 3 (REIS, 2003b). Essas novas características influenciam o processo de desenvolvimento de software que é utilizado para produzir um software livre, trazendo práticas que são diferentes dos modelos de engenharia de software tradicional. Frente ao sucesso dos projetos de SL[39], pesquisadores têm investigado essas práticas, que poderão ser incorporadas pelo desenvolvimento de software tradicional e adotadas por empresas (Seção 2.5.4).

Tabela 3: Características marcantes do desenvolvimento de software livre (REIS, 2003b)

Característica	Descrição
Distribuição via Internet	O software é disponibilizado por meio de algum serviço Internet, incluindo páginas na Web e servidores FTP, repositórios CVS e correio eletrônico.
Desenvolvimento descentralizado via Internet	Se mais de um desenvolvedor trabalha no software, este desenvolvimento é feito de maneira colaborativa, usando a Internet como meio de comunicação.
Usuários participantes	É comum formarem-se grupos de usuários finais que se comunicam com alguma regularidade com os desenvolvedores e entre si, comunicando problemas e trocando experiências do uso do software.
Interesse pessoal do autor	O autor é um usuário do software e tem motivação pessoal na sua criação e manutenção. Além disso, o autor normalmente busca apoiar seus usuários e fazer crescer o grupo de pessoas envolvidas com o software.
Ferramentas de comunicação e de desenvolvimento distribuído	Diversas ferramentas e serviços são utilizados para suportar a comunicação entre estes indivíduos.
Forte individualismo	Tanto desenvolvedores quanto usuários tratam-se pessoalmente, usando nomes próprios e não em termos da organização que possam representar; não há senso de hierarquia, neste sentido (embora haja outras hierarquias relevantes). No pacote de distribuição do software, são incluídos os nomes dos autores e contribuintes.

Raymond (1999) propôs a adoção do Modelo Bazar para o desenvolvimento de software livre. O modelo Bazar já contemplava boa parte das características identificadas por Reis

[38] http://www.frameworkdemoiselle.gov.br

[39] Exemplos de projetos de sucesso em software livre são: o Open Office (http://www.openoffice.org/) e as distribuições do linux (http://www.ubuntu.com/).

(2003b), tais como: disponibilização de código fonte; envolvimento de diversos desenvolvedores, em geral, voluntários; dispersão geográfica dos mesmos em um processo colaborativo; tempo curto de desenvolvimento; e liberdade para os desenvolvedores escolherem o trabalho que querem realizar (SILVA e FALBO, 2006). O Modelo Bazar foi uma primeira iniciativa de descrever o processo de desenvolvimento do software livre, ainda que informalmente.

Silva e Falbo (2006) apresentam uma definição de processo para desenvolvimento de software livre mais formalmente e afirmam não terem encontrado trabalhos anteriores que trouxessem essa definição. Ainda segundo Silva e Falbo (2006), apenas foram encontrados alguns trabalhos que começavam a investigar o assunto, como: Nakagawa (2004) e Reis (2003b). Entenda-se por "processo para desenvolvimento de software livre", tão somente o processo que reflete o fluxo de atividades utilizado pelas comunidades para desenvolver software livre. Ou seja, não necessariamente, o processo proposto pelos autores é um processo livre, que possa ser adaptado e reutilizado, apesar de essa possibilidade ser analisada por eles.

A expressão **processo livre para desenvolvimento de software** será utilizada para designar um processo de desenvolvimento que é disponível, e pode ser alterado ou adaptado. Para esse processo, todos os componentes deverão ser igualmente livres, inclusive as ferramentas de apoio à execução do processo. Um processo livre para desenvolvimento de software poderá ser utilizado para desenvolvimento de software livre ou proprietário.

2.5.3 O projeto Qualipso

O projeto integrado Qualipso (*Quality Platform for Open Source*) propõe-se a definir e implementar tecnologias, procedimentos (processos), leis e políticas com o objetivo de potencializar as práticas de desenvolvimento de software livre, tornando-as confiáveis, reconhecidas e estabelecidas na indústria (QUALIPSO, 2005). Para viabilizar o projeto e a sustentação do software livre como uma solução confiável para a indústria, um consórcio formado por indústrias, academia e governo foi criado. O projeto conta com a participação de colaboradores de diferentes origens: França, Itália, Brasil, Espanha, China, Alemanha e Escócia.

A abordagem do Qualipso é aberta no sentido de: (a) utilizar padrões abertos e desenvolver software de código aberto; (b) ser baseado em uma comunidade aberta, formada por cientistas, pesquisadores, profissionais e usuários, para evoluir seus recursos; e (c) ser aberto a expansões por meio da inserção de novos cenários de aplicação e de resultados de projetos.

Os objetivos estabelecidos para o projeto são, entre outros (QUALIPSO, 2005):

1. Definir métodos, processos de desenvolvimento e modelos de negócio para desenvolver e implementar sistemas em SL, garantindo aos consumidores de software que determinado projeto de SL está em conformidade com os padrões exigidos pela indústria;
2. Implementar e oferecer apoio a práticas de gerenciamento de informações (incluindo código fonte, documentação e troca de informações entre os envolvidos em um projeto de software) para melhorar a produtividade do desenvolvimento e a evolução de soluções em software livre;
3. Desenvolver uma rede de profissionais duradoura que se preocupe com a qualidade das soluções desenvolvidas em software livre para as empresas de computação.

O Qualipso é estruturado em duas classes de atividades: (a) atividades problema e (b) atividades de projeto (QUALIPSO, 2005). A associação entre as atividades de projeto, as atividades problema e as fases do projeto é ilustrada na Figura 13 [40].

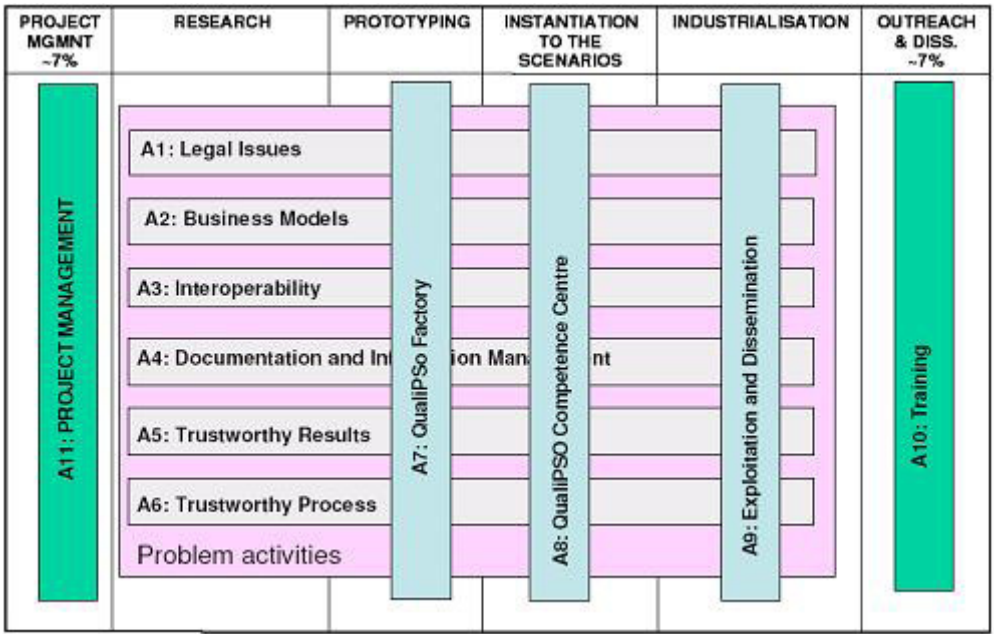


Figura 13: Estrutura do projeto Qualipso. Extraído de Qualipso (2005)

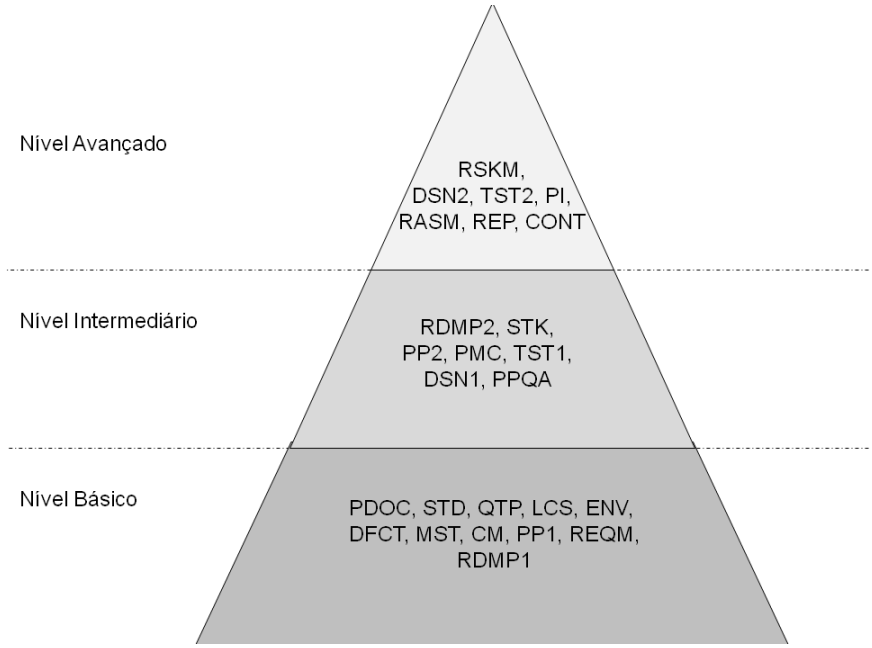
As atividades problema são as grandes questões que o projeto tem que responder. Elas fornecem as fundações nas quais o projeto será construído. Este trabalho de doutoramento está significativamente relacionado à atividade problema **A6: Processo confiável** (*Trustworthy Process*). São objetivos do projeto Qualipso, relacionados a esta atividade: (i) definir um modelo de maturidade para software livre inspirado no CMMI por meio da identificação de fatores que podem afetar a confiabilidade do desenvolvimento de SL; e (ii) definir especificações para

[40] O detalhamento das atividades do Qualipso está disponível em http://qualipso.icmc.usp.br/.

implementar processos confiáveis de software livre, com uso do ambiente colaborativo desenvolvido pelo projeto Qualipso.

Qualipso Open Source Maturity Model - OMM

Segundo Qualipso (2005), apesar de existirem diversos repositórios de informação, existe uma grande lacuna na organização do conhecimento associado ao software livre, que tipicamente é utilizado de forma disjunta pelas diversas comunidades. Organizar esse conhecimento é uma das metas do Qualipso. Convergindo para organizar esse conhecimento, um dos principais objetivos do projeto é desenvolver um modelo para software livre, inspirado no CMMI, que identifique um conjunto de práticas que devem ser aplicadas para alcançar um desenvolvimento de software livre confiável (Figura 14). Tal modelo está em construção e possui o nome temporário de OMM (*Qualipso Open Source Maturity Model*) [41] (WITTMANN et al., 2009).



Legenda:

RSKM – Risk Management
DSN2 – Design 2
STK – Stakeholders
PMC – Project Monitoring and Control
PP1 – Project Planning 1
REQM – Requirements
DSN 1- Design 1
TST2 – Testing 2
PI - Product Integration
PP2 – Projetct planning 2
PPQA – Product and Project Quality Assurance
LCS – License
RDMP1 – Roadmap 1
RASM - Results of assessment
REP - Reputation
TST1 – Testing 1
MST – Maintainability and stability
ENV – Environment
CM – Configuration management
CONT - Contributions
RDMP2 – Roadmap 2
STD – Standards
DCFT – Number of Commits and Bug Reports
PDOC – Product Documentation
QTP – quality of Test Planning

Figura 14: Níveis de maturidade do OMM. Adaptado de Wittmann et al.(2009)

A autora desta tese é uma das autoras do OMM e contribuiu para definir o modelo, baseando-se nos estudos e contribuições desta tese. Por ser um trabalho essencialmente colaborativo, é difícil delimitar as contribuições específicas de cada autor para o OMM. Algumas

[41] http://qualipso.icmc.usp.br/OMM/OMM_V3.pdf

contribuições mais facilmente delimitáveis serão discutidas nesta tese oportunamente. Nesta seção, é apresentada uma visão geral da versão atual do modelo.

Os objetivos específicos do OMM são:

- Prover, para a comunidade de software livre, a base para desenvolver produtos de maneira eficiente e de uma forma que inspire confiança para os clientes em potencial, inclusive para os "integradores de software livre" (empresas que querem integrar soluções de SL ao desenvolvimento dos seus próprios produtos); e
- Prover para os integradores de software livre uma base para avaliar o processo utilizado por uma comunidade desenvolvedora de software livre.

Assim como o CMMI, o OMM é organizado em níveis, cada nível complementando e incluindo elementos de confiança do nível anterior. Os três níveis do OMM (Figura 14) são: "Básico", "Intermediário" e "Avançado".

Os elementos de confiança foram identificados a partir de duas fontes: (i) áreas de processo do CMMI; e (ii) elementos de confiança de processos de software livre, identificados a partir de um questionário aplicado para a comunidade (QUALIPSO, 2008).

2.5.4 Aplicação de práticas do desenvolvimento de SL em organizações

Diferentes soluções para desenvolvimento de SL têm sido aplicadas em contextos organizacionais. Por exemplo, algumas experiências têm sido reportadas no sentido de utilizar ADC e de entender as possibilidades de formação de comunidades de colaboração para o desenvolvimento de software dentro de organizações. No entanto, o foco desses trabalhos têm sido promover um ambiente colaborativo para o desenvolvimento do software em si e não um ambiente distribuído e colaborativo para a MPS. Ainda assim, lições aprendidas com essas experiências podem ser transpostas para o contexto da MPS e, possivelmente, contribuir para um **desenvolvimento distribuído e colaborativo de processos**. Por esse motivo, aplicações de práticas do desenvolvimento de software livre em organizações são consolidadas nesta seção.

Dinkelacker et al. (2002) - definiram o *Open Source Progressive* (POS), modelo que aproveita práticas do modelo de desenvolvimento de SL em grandes organizações. Os autores sugerem, por exemplo, a criação de comunidades de desenvolvimento de software e a abertura do código-fonte do software para revisão e modificação por outros desenvolvedores, de acordo com sua classificação que varia de "código corporativo" até software livre, passando por código controlado, que é o código compartilhado apenas com organizações parceiras. Cada desenvolvedor da organização pode escolher o projeto de software para o qual contribuirá. Suas

idéias começaram a ser experimentadas na Hewlett-Packard (HP) em dois programas e algumas lições aprendidas foram apontadas, sendo a principal a constatação de que os maiores desafios encontrados estão relacionados às questões sociais e não às questões técnicas: (i) existência de lideranças que estimulem as pessoas a contribuir; e (ii) os desenvolvedores e gerentes precisam se sentir confortáveis com o compartilhamento do código para toda a organização.

Melian et al. (2002) - relatam uma pesquisa alinhada com o trabalho de Dinkelacker et al. (2002), também na HP, com o objetivo de entender e documentar um novo paradigma de colaboração na HP. No escopo do modelo POS, eles exploram a natureza das comunidades que se desenvolvem para apoiá-lo.

Gurbani et al. (2006) – exploram a possibilidade de conduzir o desenvolvimento de um software no contexto organizacional, seguindo a dinâmica de desenvolvimento de software livre. Eles apresentam um estudo de caso, algumas lições aprendidas e principalmente mostram como o assunto ainda envolve várias questões de pesquisa em aberto.

Riehle et al. (2009) – propõem a utilização de um ADC, uma adaptação do Gforge[42], em uma organização para apoiar o desenvolvimento de software. A utilização de uma adaptação do Gforge apoiaria a execução de algumas práticas de software livre no ambiente organizacional. Segundo os autores, as organizações podem beneficiar-se significativamente de práticas de software livre, em especial do uso de voluntários para o desenvolvimento, que podem complementar a alocação tradicional e hierárquica de recursos. Tais voluntários se mostraram motivados e contribuíram em diferentes fases como requisitos e teste.

Experiências utilizando trabalhos cooperados começam a ser desenvolvidas no SERPRO.

2.6 Considerações finais

Neste capítulo, foi apresentada uma síntese das principais pesquisas que estão sendo conduzidas e que serviram de base para a condução das atividades referentes ao presente trabalho de doutorado. Considerando-se a natureza da linha de pesquisa abordada nesta tese, uma diversidade de temas precisou ser investigada. Por meio da revisão conduzida, podem-se identificar diversas linhas de pesquisa ainda em aberto e que merecem investigação, em especial nas oportunidades de interseção entre essas áreas de pesquisa.

Várias pesquisas enfocam a MPS como uma maneira de aumentar a qualidade do produto de software e/ou reduzir o tempo e o custo de desenvolvimento, associando a qualidade do

[42] O GForge é um ADC baseado SourceForge versão 2.1 e disponibilizado como SL. http://gforge.org/gf/

produto à qualidade do processo adotado para produzi-lo. As principais abordagens propostas na literatura para gerenciar a adoção sistemática de MPS foram apresentadas. Essas abordagens têm em comum a repetição de um conjunto de ciclos e o enfoque no aprendizado com base na experiência obtida em ciclos anteriores. A maneira como a MPS deve ser concretizada, no entanto, não é detalhada nessas abordagens. Mecanismos e estratégias que as viabilize merecem ser investigados. Dada a carência de trabalhos na área e, ao mesmo tempo, a relevância do tema, identificou-se outras iniciativas para definir estratégias e mecanismos de apoio à MPS em paralelo com o desenvolvimento deste trabalho de doutorado (SANTOS et al. 2007; MONTONI et al., 2007). Essas iniciativas são complementares a ColabSPI.

Além disso, os autores (SANTOS et al., 2007, MONTONI et al., 2007) identificaram alguns fatores críticos de sucesso e cinco lições aprendidas para o sucesso da MPS. O aprendizado dos autores foi considerado como insumo para a organização dos fatores críticos de sucesso do Capítulo 4.

As tendências de uso de GC para a engenharia de software foram sintetizadas. A investigação dessa temática foi motivada pelo seu potencial de contribuição para a MPS.

A revisão conduzida permitiu, também, a identificação de pesquisas iniciais relacionadas a processos de desenvolvimento de software livre. O desenvolvimento de software livre tem algumas características semelhantes ao desenvolvimento distribuído. Ainda assim, vários problemas estão em aberto na conexão entre as áreas de pesquisa em processos de software e em software livre. Iniciativas recentes indicam a relevância do tema, sendo exemplos o EPF e, principalmente, o Qualipso. No projeto Qualipso, está sendo desenvolvido um modelo de maturidade para abordar as questões específicas relacionadas a processos de desenvolvimento de software livre. A autora deste trabalho participa do projeto Qualipso, em especial nas atividades relacionadas a processos confiáveis (A6).

O conhecimento em desenvolvimento de SL, associado ao conhecimento em MPS e GC, foi fundamental para:

- Propor a estratégia distribuída e colaborativa para a MPS, a partir do entendimento das características do desenvolvimento de software livre, em especial suas estratégias de colaboração e comunicação;
- Iniciar a definição da evolução do processo Demoiselle (Capítulo 7);
- Contribuir como co-autora para a definição do OMM;
- Organizar recomendações para a MPS no âmbito do desenvolvimento de SL;
- Entender as necessidades e contribuir para os centros de competência em software livre do Qualipso, em particular dos centros de competência do Instituto de

Matemática e Estatística da USP (CCSL-IME) e do Instituto de Ciência da Computação e Matemática Computacional da USP (CCSL-ICMC); e

- Elaborar e aplicar questionários para avaliar o OMM e caracterizar projetos de SL.

3- Fatores que influenciam os resultados da MPS

Neste capítulo, é apresentada uma análise de fatores que podem influenciar os resultados de programas de MPS. A esquematização de tais fatores pode ser útil para empresas que estejam conduzindo programas de MPS e também para pesquisadores em engenharia de software. A referida esquematização foi utilizada para identificar os requisitos da ColabSPI (Capítulo 5).

Este capítulo está assim organizado: na Seção 3.1, algumas considerações iniciais são discutidas. Na Seção 3.2, são descritos os passos seguidos para entender as influências à MPS e são referenciados os artigos considerados mais relevantes entre os encontrados durante a busca por fatores/ motivadores e barreiras para a MPS. Na Seção 3.3, um esquema organizado de fatores[43] para a MPS é apresentado. Na Seção 3.4, é discutida a relação entre a MPS e suas influências e o desenvolvimento de software livre. Finalmente, na Seção 3.5, são apresentados comentários acerca da análise de fatores de impacto na MPS.

3.1 Considerações iniciais

Entender como implementar a MPS com sucesso é uma tarefa complexa. A fim de propor uma estratégia de MPS colaborativa e distribuída, foi necessário compreender fatores que afetam a MPS e como mudanças em processos devem ser refletidas na organização. A percepção do ambiente e da trajetória interna das organizações na execução dos seus processos de MPS (observação da MPS) pode: (i) fornecer pistas valiosas acerca das causas do baixo rendimento ou aderência a processos de software; (ii) ajudar a prever os efeitos das iniciativas de MPS e estabelecer as bases para definir uma estratégia de MPS e seus critérios de avaliação.

Foram analisados diversos estudos que analisaram iniciativas de MPS para identificar influências para o sucesso (ou fracasso) de programas de MPS. Além disto, lições de MPS foram obtidas a partir da observação em campo de unidades de desenvolvimento de software em uma

[43] Não foi encontrado um termo comum para referir-se a influências para o sucesso (ou fracasso) da MPS. São utilizados “influências”, “fatores”, “barreiras” e “motivadores”. O termo “fator” será adotado como padrão.

organização de grande porte. As lições aprendidas com a observação em campo são comentadas no Capítulo 4. As observações em campo foram baseadas em doze unidades de desenvolvimento desafiadas a melhorar continuamente seu processo de desenvolvimento de software e seu nível de maturidade alinhado ao CMMI e às orientações corporativas. As descobertas foram baseadas na análise do conteúdo de relatórios de avaliação, discussões/reuniões dos grupos de processo de engenharia de software e opiniões de desenvolvedores.

3.2 Identificação de fatores de influência para MPS

Os principais passos para analisar os fatores com impacto na MPS são apresentados na Figura 15.

Figura 15: Principais passos executados para analisar fatores para a MPS

Para identificar artigos sobre fatores que influenciam a MPS foi estabelecido um protocolo de revisão sistemática, contendo o seguinte critério de busca: *("software process improvement" or SPI) and (barrier or motivator or influence or enabler or inhibitor or "success factor")*. Mais de 1000 artigos foram identificados atendendo ao critério de busca nas seguintes fontes: ACM Digital Library, IEEE Explore e Science Direct. O número elevado de artigos é justificado pela abrangência dos termos da pesquisa. A leitura dos títulos e resumos permitiu a seleção dos artigos que, de fato, discutiam fatores de influência para programas de MPS. Ao

selecionar apenas aqueles que apresentavam resultados experimentais, o número de artigos foi reduzido a menos de 30. Um extrato representativo desses artigos é apresentado na Tabela 4.

Tabela 4: Resumo dos principais artigos sobre fatores para a MPS

Goldenson e Herbsleb (1995) conduziram uma das primeiras entrevistas para avaliar diversos fatores organizacionais que pudessem influenciar a MPS. Foram entrevistados 138 indivíduos de 56 organizações nos Estados Unidos da América e no Canadá. Os fatores identificados com significância estatística foram incluídos nesta pesquisa.
Stelzer e Mellis (1999) conduziram um estudo exploratório de fatores que podem afetar a mudança organizacional na MPS e analisaram relatórios contendo experiências de 56 empresas que tinham ou implementado um programa de qualidade ISO 9000, ou conduzido uma iniciativa de MPS inspirada no CMM. A partir deste estudo, eles identificaram 10 fatores em potencial, que foram incluídos nesta pesquisa.
El-Emam et al. (2001) identificam e priorizam fatores que podem afetar os resultados da MPS, descrevendo como alguns desses fatores podem estar relacionados. Os resultados são obtidos a partir de questionários aplicados em organizações que haviam passado por avaliações baseadas no CMM-SW há algum tempo, as mesmas organizações estudadas por Goldenson e Herbsleb (1995). Com outros autores, El-Emam havia, anteriormente, analisado organizações que passaram por avaliação baseada na ISO/IEC 15504 (EL-EMAM et al., 1999)
Dyba (2001) desenvolveu um instrumento para medir fatores críticos para o sucesso da MPS com base em dados coletados em 120 organizações de software. Ele provê uma síntese das recomendações para uma efetiva gestão da qualidade e da MPS encontradas na literatura e confirmadas por estudos experimentais. O resultado deste estudo foi um conjunto de seis fatores que possibilitam o sucesso em MPS.
Baddoo e Hall (2002) apresentam resultados baseados em estudos experimentais em 13 empresas de software, nos quais foram conduzidos grupos de entrevista (*focus groups*) com aproximadamente 200 profissionais. Eles buscaram melhor compreender como empresas podem maximizar o apoio dos profissionais à MPS e as possíveis associações entre os motivadores da MPS. Os autores elaboraram outros dois artigos que analisam o resultado dos mesmos estudos experimentais sob diferentes perspectivas. Baddoo e Hall (2003) exploram os desmotivadores para MPS e Baddoo et al. (2007) analisaram a relação entre os desmotivadores da MPS, sob o ponto de vista dos papéis: desenvolvedores, gerentes seniores e líderes de projeto.
Rainer e Hall (2002) relatam os resultados de um questionário para apurar fatores chave que impactam a MPS. Eles fazem referência ao trabalho de Goldenson e Stelzer (1995) apud Rainer e Hall (2002). Eles concluem que organizações em diferentes níveis de maturidade consideram diferentes fatores críticos como tendo um grande impacto na MPS.
Niazi et al. (2005a) identificaram fatores críticos para o sucesso da MPS. Eles comparam os seus resultados com aqueles da literatura e ordenam os fatores de acordo com a sua ocorrência tanto na literatura quanto em entrevistas. Diversos artigos sobre MPS foram publicados pelos autores. Para uma visão geral da sua pesquisa, consultar Niazi et al. (2005b).
Montoni e Rocha (2007) sintetizaram os resultados de estudos experimentais com o objetivo de identificar quais os fatores que influenciam a MPS. Adicionalmente, conduziram um inquérito com profissionais envolvidos em experiências de MPS na indústria brasileira de software. Suas descobertas reforçam alguns dos fatores críticos para o sucesso citados na literatura.
Nasir et al. (2008) discutem barreiras (*resistance factors*) na implantação de projetos de MPS na Malásia envolvendo 20 empresas e 174 participantes. A falta de participação dos desenvolvedores na MPS foi o mais crítico fator identificado. Outro fator de interesse mencionado pelos autores, que tem implicações diretas em uma MPS em ambiente distribuído, é a influência de questões culturais na adoção de modelos de maturidade em diferentes países.
Coleman e O'Connor (2008) apresentam os resultados de um estudo sobre como o processo de desenvolvimento de software e a MPS são aplicados na prática usando a indústria irlandesa de produtos software como exemplo. O estudo utilizou a metodologia da teoria fundamentada (*grounded theory*) para produzir uma teoria, baseada em dados de campo, que explica como os processos de software se formam e evoluem e, quando e porque a MPS é realizada. Suas descobertas também ajudam na análise dos fatores da MPS.

Mais de uma centena de fatores/motivadores (ou barreiras) para a MPS foram extraídos dos artigos selecionados na revisão. Para identificar os fatores relacionados, cada fator foi listado

isoladamente como uma única influência e representado em um pequeno papel (*post-it*). Todos os *post-it* foram colocados lado a lado. Foram acrescentados *post-it*, referentes a fatores observados em campo, extraídos da experiência relatada no Capítulo 4.

3.3 Analisando relações identificadas entre fatores da MPS

Embora muito útil, apenas identificar as influências não foi suficiente para definir uma estratégia para apoiar a MPS. Uma organização de tais influências foi necessária, indicando como elas relacionam-se umas às outras; como elas podem ser generalizadas; e quais seus diferentes níveis de abstração e de importância relativa.

Nesse sentido, o próximo passo na análise foi agrupar fatores (*post-it*) com significado similar ou aproximado. Grupos de fatores foram naturalmente tornando-se evidentes. Fatores positivos e barreiras que afetam a MPS foram convertidos para uma forma neutra, a fim de facilitar o seu relacionamento e identificar redundâncias. Foi possível eliminar casos de redundância evidente (aproximadamente 13%).

Para cada grupo, um fator representativo foi selecionado, que melhor consolida o assunto daquele grupo, e de fatores relacionados (coluna da esquerda, na Figura 16). Cada grupo é representado por uma linha. Os grupos de fatores foram organizados em categorias. O esquema de organização utilizado para cada uma das categorias identificadas é ilustrado na Figura 16. As caixas em roxo são fatores observados em campo, não claramente redundantes quando comparados com fatores da literatura.

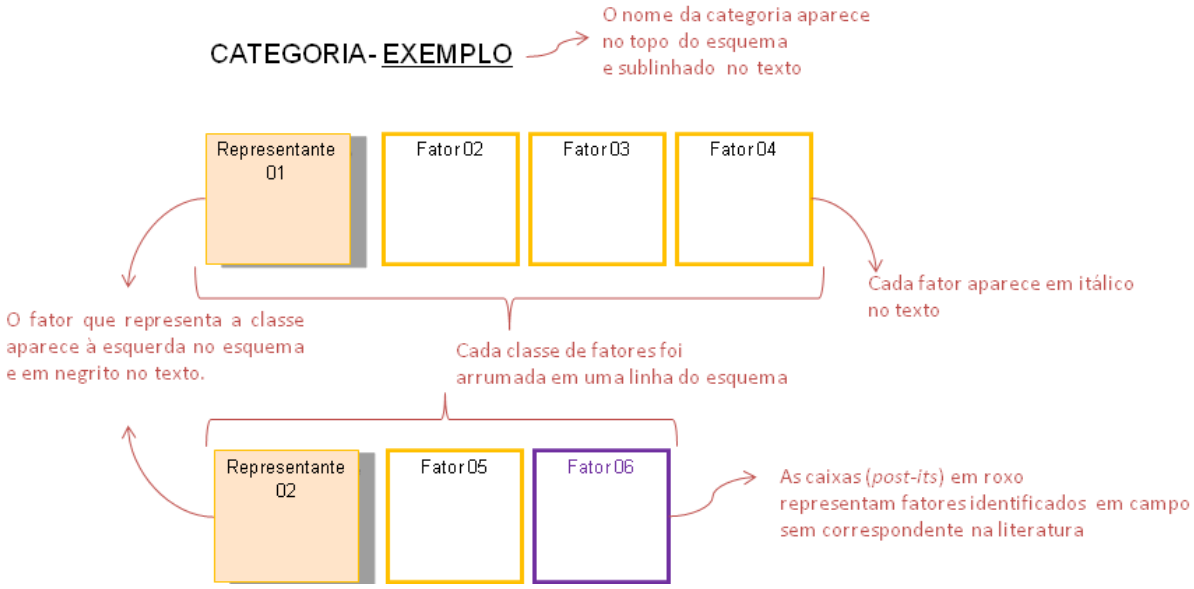


Figura 16: Notação para o esquema de fatores de influência à MPS

Todos os grupos de fatores foram arranjados em três categorias principais, consideradas fundamentais para o efeito "desempenho do processo de desenvolvimento de software", que redunda do desempenho da MPS:

- **"**ASPECTOS ORGANIZACIONAIS**"** – seus fatores estão relacionados ao potencial do ambiente organizacional em suportar iniciativas de MPS (Seção 3.3.1);
- **"**MELHORIA CONTÍNUA**"** – seus fatores estão relacionados à necessidade do processo de desenvolvimento de software sofrer melhoria contínua e adaptações que podem afetar o seu desempenho (Seção 3.3.2); e
- **"**ADERÊNCIA**" –** seus fatores estão relacionados a: (i) adequação ou habilidade para executar o processo; (ii) uso de um processo definido; e (iii) supervisão, comparação (*benchmarking*) e avaliação do processo (Seção 3.3.3). Segundo Florac (1997), estes três itens podem significativamente afetar o desempenho de um processo de desenvolvimento de software. Questões relacionadas à aderência são: O processo está sendo fidedignamente executado? Os projetos estão comportando-se de acordo com os padrões propostos no processo?

Diagramas de causa e efeito foram usados para explorar causas potenciais ou reais (entradas) que resultariam em um único efeito (saída). Novos esquemas foram estruturados, desta vez buscando estabelecer relações de causa e efeito entre eles.

Os fatores foram arranjados de acordo com o seu nível de abstração, resultando em uma descrição de relações entre eles e uma hierarquia de fatores. Tal arranjo pode auxiliar usuários a procurar por fatores que sejam causas de outros fatores, a identificar áreas problemáticas e comparar a importância relativa dos diferentes fatores. Diferentes níveis de abstração foram identificados. Alguns fatores eram gerais, como, por exemplo, "**Política**"; enquanto outros eram mais específicos, como "*Realização de auditorias externas*". Em função disto, alguns fatores foram explorados com maior profundidade em novos diagramas de causa e efeito. Embora um diagrama de causa e efeito permita a representação de múltiplos níveis hierárquicos, decidiu-se utilizar múltiplos diagramas para representar diferentes níveis de abstração. Esta abordagem permite que se obtenha primeiro uma visão geral do problema (Figura 17) e, em seguida, algumas das causas podem ser revisitadas e detalhadas em novos diagramas de causa e efeito. A cor azul indica grupos de fatores que são detalhados posteriormente em novas categorias.

As causas destacadas em verde e azul (Figura 17) estão intimamente relacionadas com a estratégia proposta (Capítulo 5). As causas em cinza estão além do escopo deste trabalho.

Arranjar os fatores considerando as três categorias "MELHORIA CONTÍNUA", "ADERÊNCIA" e "ASPECTOS ORGANIZACIONAIS" pode facilitar o tratamento de tais fatores, dividindo o trabalho da MPS em iniciativas mais relacionadas ao SEPG (*Software Engineering Process Group*), às unidades de desenvolvimento e à gerência organizacional, respectivamente. No entanto, percebeu-se também a conveniência de organizar mais duas categorias, relacionadas a: "PARTICIPAÇÃO E MOTIVAÇÃO" e "COMUNICAÇÃO" já que fatores agrupados nas duas novas categorias se relacionam tanto com "MELHORIA CONTÍNUA" quanto com "ADERÊNCIA". As cinco categorias são apresentadas nas subseções seguintes.

Figura 17: Diagrama geral de fatores que podem influenciar a MPS

3.3.1 Categoria Aspectos Organizacionais

Os fatores da categoria "ASPECTOS ORGANIZACIONAIS" (canto inferior esquerdo, Figura 17) estão relacionados ao ambiente organizacional, no qual o programa de MPS está embutido. Observe-se que o ambiente organizacional é mais abrangente que a "organização para MPS", que é a organização específica, dentro do ambiente organizacional, para tratar as atividades de MPS. Os fatores dessa categoria representam as condições para iniciar e sustentar

iniciativas de MPS. A MPS só pode ser realizada com sucesso se os gestores e os funcionários têm uma compreensão aprofundada do ambiente organizacional como um todo. Uma melhor compreensão incentiva os engenheiros de software participar e conduzir iniciativas de MPS alinhadas aos objetivos estratégicos.

Os aspectos organizacionais não são o foco da estratégia para a MPS proposta, que é mais focada nos fatores relacionados à melhoria contínua em si e à aderência. Por conseguinte, esta categoria não será apresentada de forma detalhada. Ainda assim, sete grupos principais foram identificados no âmbito da categoria e estão ilustrados na Figura 18. Todos os fatores observados em campo foram mencionados na literatura revisada, com exceção de "*rotatividade da alta direção*" (caixa roxa na Figura 18). A observação desse fator em campo pode ter acontecido porque a rotatividade de membros da alta administração é um fator de especial importância em organizações governamentais.

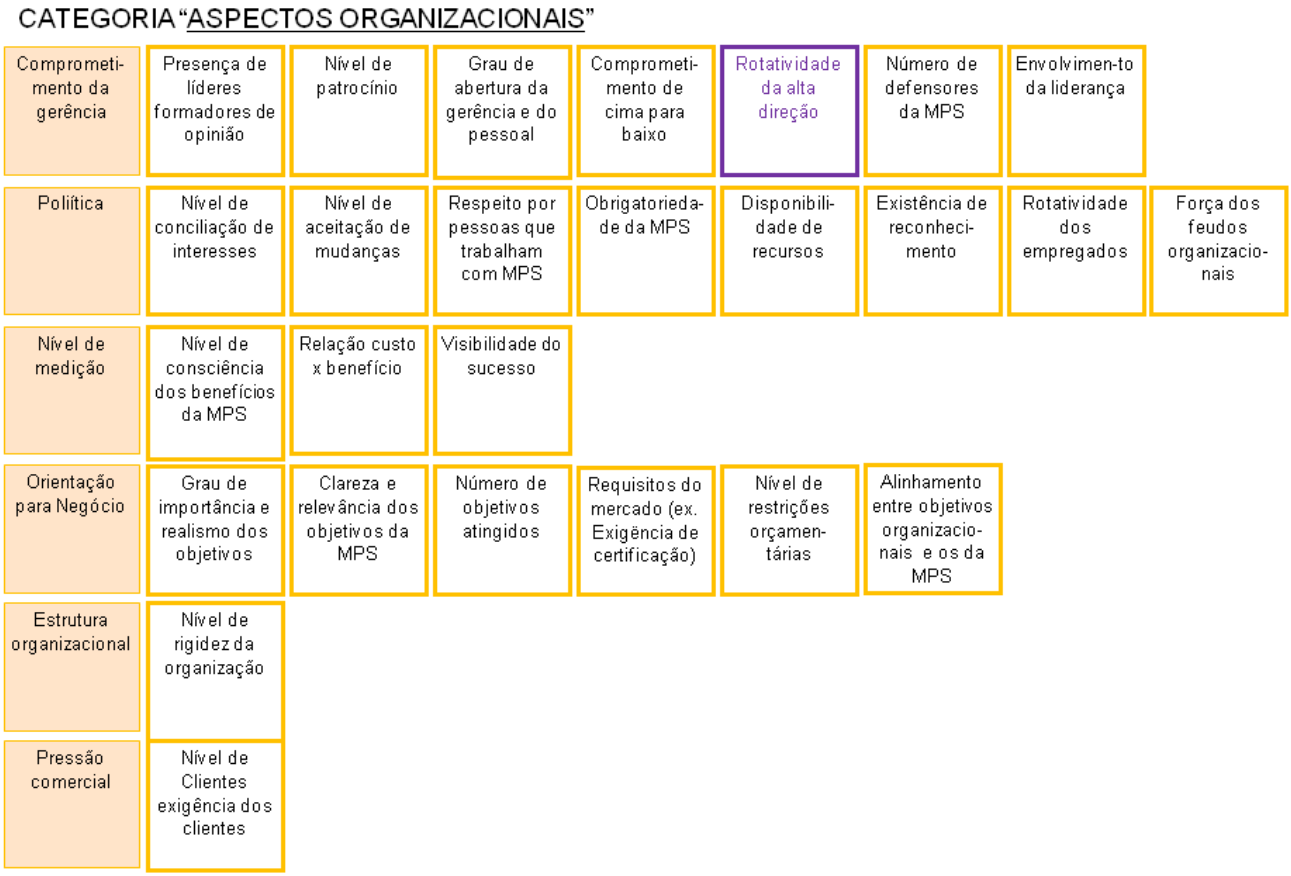


Figura 18: Fatores da categoria "Aspectos Organizacionais"

Mais detalhes relacionados com a categoria "ASPECTOS ORGANIZACIONAIS" podem ser encontrados, por exemplo, em: El-Eman et al. (2001); Beecham et al. (2002); Nasir et al. (2008) e Baddoo et al. (2007).

3.3.2 Categoria Melhoria Contínua

A categoria "MELHORIA CONTÍNUA" agrupa os fatores relacionados com a definição e evolução do processo de desenvolvimento de software e as estratégias para implementar as melhorias nesse processo. Esta categoria está focada no planejamento e controle das iniciativas de MPS; na definição de uma estratégia de MPS; na decisão sobre as melhorias; e na adequação do processo de desenvolvimento de software. Cinco grupos principais foram identificados para esta categoria (Figura 19).

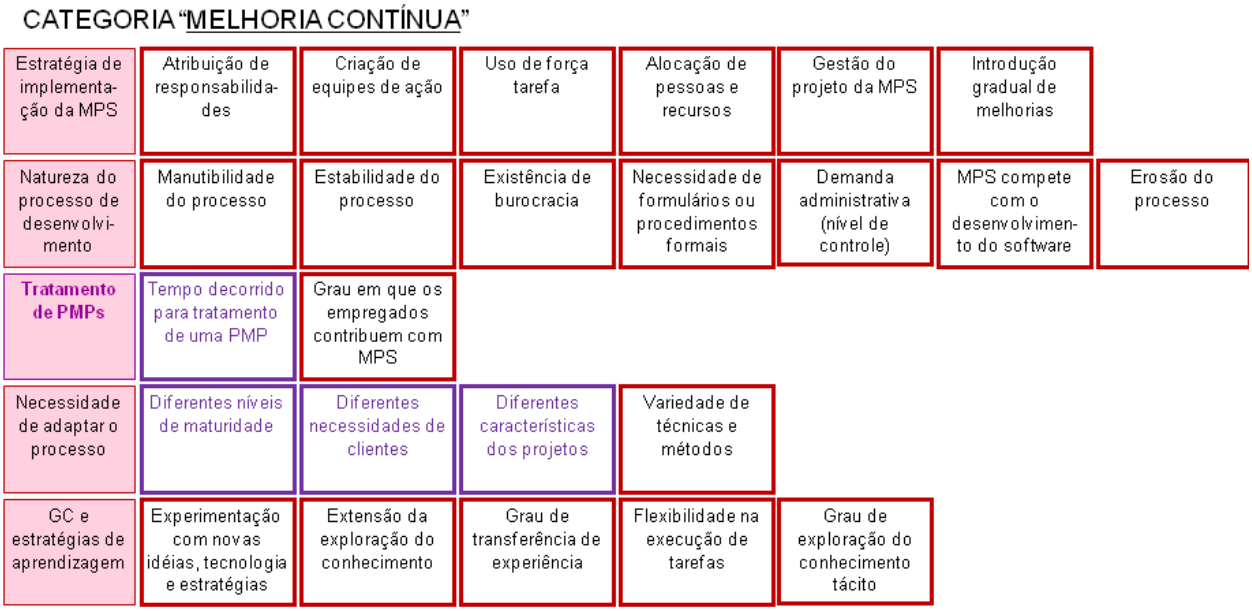


Figura 19: Fatores da categoria "Melhoria Contínua"

O grupo "**estratégia de implementação da MPS**" (primeira linha, Figura 19) desdobra as metas da organização para a MPS em diferentes iniciativas para atingi-las. Cada uma dessas iniciativas pode ser vista como um fator que pode favorecer o sucesso da MPS. Fatores neste grupo estão relacionados a: (i) estrutura organizacional de MPS; (ii) estratégias para otimizar a alocação de recursos; e (iii) organização iterativa das iniciativas da MPS. Em relação aos fatores relacionados à estrutura organizacional da MPS, foi reportado que a *criação de equipes de ação*[44] para MPS (NIAZI et al., 2005a; EL-EMAN et al., 1999) e o *uso de força tarefa* (BADDOO e HALL, 2002; EL-EMAN et al., 1999) para conduzir a MPS são fatores que a influenciam positivamente.

Os fatores *alocação de pessoas e recursos* e *atribuição de responsabilidades* (GOLDENSON e HERBSLEB, 1995) estão relacionados a aspectos organizacionais (fator

[44] O CMMI menciona a criação de equipes de ação (*process action teams*): uma equipe que tem a responsabilidade de desenvolver e implementar atividades de melhoria de processos para uma organização, conforme documentado no plano de ação de melhoria de processos (SEI, 2006).

disponibilidade de recursos (BADDOO et al., 2007), Figura 18). No entanto, a *atribuição de responsabilidades* refere-se aqui a um controle mais específico de recursos sob gestão da MPS (como organizar e utilizar melhor os recursos já disponíveis). Uma vez disponibilizado um recurso (fator organizacional) para um gerente de MPS, este deve buscar otimizar a atribuição de responsabilidades, assim melhorando a utilização dos recursos disponibilizados.

Diferentes estratégias de implementação da MPS podem levar a resultados diferentes e facilitar ou impor barreiras. Ao compreender a dinâmica das mudanças que estão sendo propostas pela MPS, e planejar cuidadosamente a sua introdução, a administração pode minimizar o estresse para todos os envolvidos. Nesse contexto, a *introdução gradual* de mudanças (BADDOO e HALL, 2002) pode ser um fator positivo para a MPS.

A "**natureza do processo de desenvolvimento de software**" (Linha 3, Figura 19) também traz alguma complexidade para os esforços de MPS. O processo de desenvolvimento de software deve ser mutável e manutenível. Em outras palavras, eles devem poder ser mantidos, desenvolvidos, ou evoluídos. Assim, *manutenibilidade* e *estabilidade* são características importantes desse processo. Perseguir a ambas traz desafios extras para a MPS.

Um efeito colateral da MPS (em particular da evolução de processo) é a *erosão do processo* motivada pelo alto custo da melhoria ou pela baixa adesão dos colaboradores ao processo (COLEMAN e O'CONNOR, 2008). Outro aspecto relacionado à "**natureza do processo de desenvolvimento de software**" é que, muitas vezes, quando se tenta padronizar procedimentos ou descrever atividades, a *burocracia do processo* de desenvolvimento de software aumenta ou o processo se torna muito pesado (BADOO e HALL, 2007). Eliminar a burocracia e reduzir *formulários ou procedimentos formais* (documentação) são oportunidades de melhoria. Menos burocracia pode melhorar o desempenho do processo de desenvolvimento de software, economizando energia para as atividades principais. Coleman e O'Connor (2008) afirmam que muitas organizações estão se beneficiando de uma comunicação mais informal. Por outro lado, cabe a ressalva de que minimizar a documentação e focar apenas no *conhecimento tácito*, que é o *know-how* intuitivo e não documentado do indivíduo ou da equipe, pode ter suas limitações e pode ter custo próprio (COLEMAN e O'CONNOR, 2008). Isto pode ser especialmente verdadeiro para as empresas que estão usando o XP (BECK, 1999) e que se preocupam com a ênfase no informal (COLEMAN e O'CONNOR, 2008). Um caso especial de aumento de burocracia é o aumento da *demanda administrativa*, que pode ser mais uma barreira para programas de MPS.

O próximo grupo de fatores relacionados à MPS é "**tratamento de PMPs**". À medida que profissionais utilizam processos de desenvolvimento de software, propostas de melhoria de processo (PMPs) surgem. Uma PMP pode ser o primeiro passo para iniciar mudanças no processo. Analisar tais PMPs e processá-las requer um fluxo de atividades para receber, analisar, priorizar, desenvolver e implantar propostas em uma nova versão do processo (MALHEIROS et al., 2008). Stelzer e Mellis (1999) identificaram o grau em que uma PMP é efetivamente planejada e controlada como um fator chave para o sucesso da MPS. O tratamento de PMPs pode apoiar o planejamento e de controle da MPS, auxiliando na sua gestão. Há muitos desafios relacionados com o tratamento de PMPs, especialmente relacionados à: *tempestividade no tratamento das PMPs*; diminuição do impacto das mudanças no ambiente; e adequação da nova versão do processo às necessidades organizacionais e dos projetos (MALHEIROS et al., 2008). As possíveis causas de atrasos no tratamento de PMPs são discutidas na Seção 4.2.3.

Dyba (2001) aponta para a *extensão em que os empregados contribuem com MPS* como um indicador do sucesso da MPS. Ainda assim, a literatura sobre os fatores críticos de sucesso carece de trabalhos que aprofundem em questões sobre o tratamento de PMP, e, em particular como equipes distribuídas podem colaborar efetivamente com propostas de melhoria.

Os fatores relacionados à "**necessidade de adaptar o processo"** de desenvolvimento de software exigem esforços adicionais de programas da MPS (quarto grupo, Figura 19). Alguns fatores podem aumentar a necessidade de adaptar o processo: *diferentes níveis de maturidade*, *características dos projetos*, *diferentes necessidades de clientes* e *variedade de técnicas e métodos*. Para Stelzer e Mellis (1999), "*adaptar as iniciativas de melhoria*" significa adaptar os esforços de melhoria para os pontos fortes e fracos específicos das diferentes equipes e departamentos. A necessidade de adaptar as iniciativas pode refletir também a demanda para adequar os processos de desenvolvimento de software aos diferentes níveis de maturidade e às necessidades de cliente distintos. Quanto maior e mais detalhado o processo, e quanto mais ele acomoda casos especiais e alternativas para permitir a adaptação, mais difícil será sua evolução.

Os *diferentes graus de maturidade* das unidades devem ser considerados antes de decidir-se sobre uma intervenção no processo. Por exemplo, usar complexas estratégias de medição para apoiar a melhoria pode ser eficaz para as unidades de maturidade elevada, mas não é viável para as unidades de baixa maturidade. Ainda em relação a este fator, Rainer e Hall (2002) mencionam que as organizações com capacidades distintas de maturidade no processo consideram diferentes fatores críticos de sucesso como influentes sobre a MPS. As *diferentes necessidades dos clientes* podem influenciar a adequabilidade do processo também. Por exemplo, as atividades exigidas por contrato podem substituir as atividades do processo padrão, em alguns casos.

Da mesma forma, *diferentes características do projeto* podem gerar a necessidade de adequar o processo. Por exemplo, se os requisitos de software são bem conhecidos e uma grande equipe está trabalhando no projeto, provavelmente, abordagem orientada por planejamento seria desejável. Por outro lado, se os requisitos são desconhecidos e uma pequena equipe de especialistas está trabalhando no projeto, possivelmente, uma abordagem mais ágil seria melhor. Assim, projetos com características diferentes demandam processos diferentes, e, conseqüentemente, intervenções de melhoria diferenciadas.

Outro grupo de fatores para a MPS é a "**gestão do conhecimento e a estratégia de aprendizagem"**. Dyba (2001a) refere-se a diferentes estratégias de aprendizagem para manter ou transformar um processo. Segundo ele, a *exploração do conhecimento existente* e a *prospecção de novos conhecimentos* são fatores chave para o sucesso da MPS. A *extensão da exploração do conhecimento* indica a medida que a organização está aprendendo com a experiência. Quanto mais uma organização sistematiza a sua experiência passada; baseia suas rotinas formais nessas experiências, e promove a transferência de experiência interna; mais a MPS pode obter sucesso. Quanto maior o *grau de transferência de experiência*, melhor. Explorar novos conhecimentos significa aprender pela experimentação. Quanto mais a inovação/criatividade é incentivada, mais a MPS tende a ter sucesso. A *experimentação com novas idéias, estratégias e tecnologias*, a *variedade na utilização de métodos, técnicas e ferramentas*, e uma grande *flexibilidade na execução de tarefas* são todos fatores que podem contribuir para o sucesso da MPS.

Em particular, documentar e distribuir o processo de desenvolvimento de software é uma estratégia de gestão do conhecimento para tornar o conhecimento sobre desenvolvimento de software explícito. O complemento do conhecimento explícito é o *conhecimento tácito*. Segundo Coleman e O'Connor (2008), por vezes, a documentação por si só não garante que todos os membros da equipe tenham um entendimento comum das necessidades do projeto e, nesse caso, o conhecimento tácito precisa ser administrado.

3.3.3 Categoria Aderência

Como mencionado, os três aspectos de "ADERÊNCIA", que podem afetar o desempenho do processo de desenvolvimento de software, são: (i) *uso real do processo na organização* (fortemente relacionadas à definição de aderência em si), (ii) a *capacidade de usar o processo*, e (iii) a *avaliação do processo*. O uso real do processo de desenvolvimento de software é fortemente influenciado pela capacidade dos desenvolvedores e gerentes para executá-lo. Em alguns casos, as pessoas estão efetivamente tentando usar o processo como definido, mas não são capazes de fazê-lo. Elas podem não ter o conhecimento adequado ou apoio técnico sobre como

seguir o processo. Outra possibilidade é que mesmo tendo conhecimento, os desenvolvedores não sigam o processo por pressões externas, como pressão por tempo, por exemplo. Fatores relacionados a essas questões são apresentados na Figura 20.

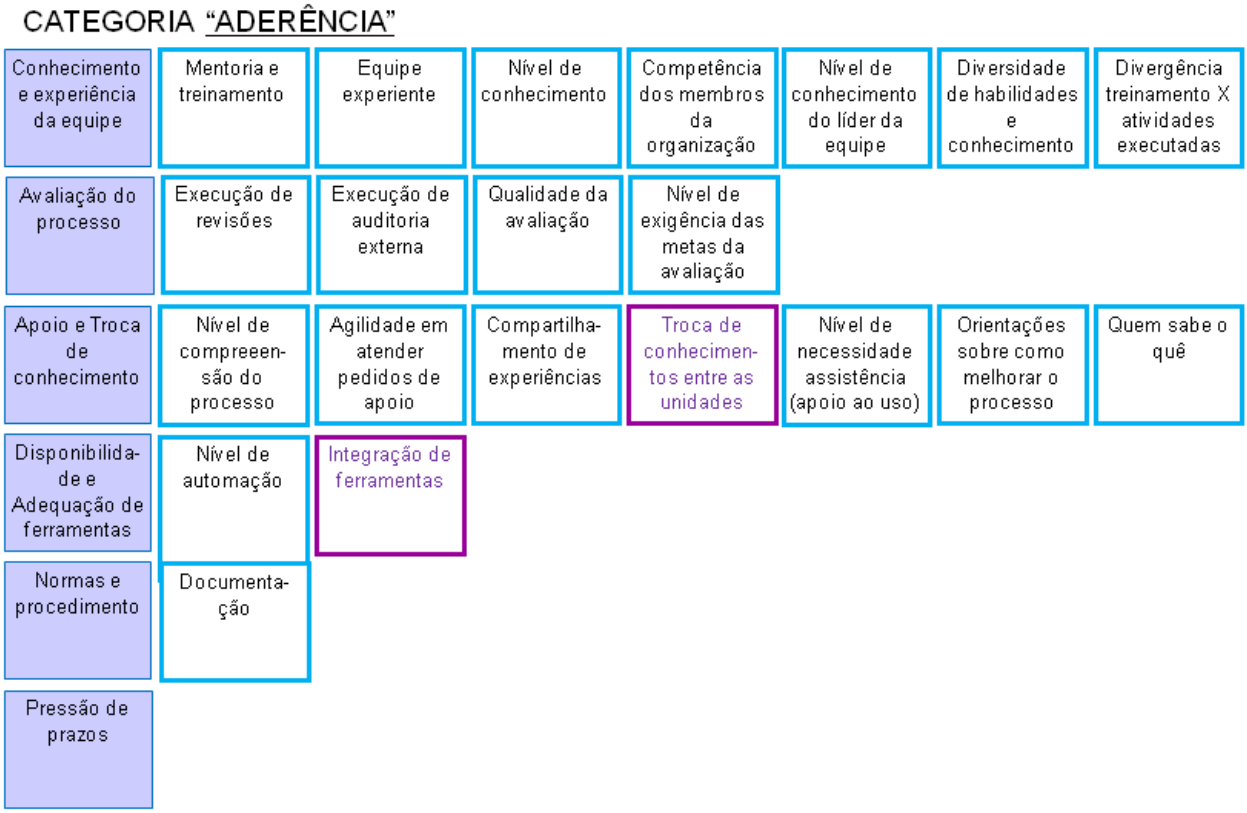


Figura 20: Fatores da categoria "Aderência"

Ter o "**conhecimento e a experiência"** (primeira linha de cima para baixo, na Figura 20) para executar o processo de desenvolvimento de software é importante para executá-lo fielmente. *Treinamento e mentoria* são fatores que podem contribuir para a aquisição deste conhecimento. No entanto, treinamentos não ajudarão se houver *divergência entre o treinamento obtido e as atividades executadas*, por exemplo, quando o treinando não tem oportunidade em curto prazo de aplicar o conhecimento adquirido. Um problema para o sucesso da MPS é que pode ser muito difícil implementar os treinamentos necessários (BEECHAM et al., 2003).

Habilidades interpessoais e o *nível de conhecimento* do desenvolvedor podem influenciar de muitas maneiras a aderência ao processo de desenvolvimento de software também. Para minimizar essas influências, uma alternativa é oferecer possibilidades de adaptar o processo de desenvolvimento de software para diferentes características de equipes, oferecendo adaptação para uma *grande diversidade de habilidades e conhecimentos*. Alguns trabalhos indicam que um alto *nível de conhecimento do líder da equipe* de engenharia de software (isto é, ele ou ela possui

experiência técnica) e a *competência dos membros da organização* são fatores importantes para a aderência ao processo. Quanto mais a *equipe é experiente*, maior a aderência.

A "**avaliação do processo"** (Linha 2, Figura 20) é um aspecto que poderá interferir na aderência ao processo. Segundo Florac (1997), todos os processos estão sujeitos a forças de entropia, isto é, se deixados sozinhos, os processos tendem a deteriorar-se de um estado controlado para um estado caótico. Avaliação ou fiscalização pode contribuir para sustentar o processo, mantendo a sua aderência e desempenho. Neste sentido *revisões, inspeções* e *auditorias externas* (BADDOO e HALL, 2002) podem ser muito importantes. Exemplos de auditorias externas são as auditorias feitas por empresas de certificação. Modelos de maturidade e modelos de avaliação associados podem ser utilizados para avaliar processos. Resultados de desempenho de organizações que adotaram o Modelo MPS e seu modelo de avaliação, por exemplo, são apresentados em Travassos e Kalinowski (2008). Apesar dos benefícios para a aderência do processo, há de se ter muito cuidado com a *qualidade da avaliação do processo*. Uma avaliação inadequada (muito rígida ou muito simplista) pode dar uma idéia equivocada do desempenho do processo ou da aderência. Isso pode afetar a aderência no futuro, porque os desenvolvedores e gerentes podem repetir comportamentos não-conformes, se não forem corrigidos por uma avaliação do processo. A adequação da avaliação do processo começa a ser estabelecida na definição das *metas para essa avaliação*. O estabelecimento de metas possíveis para a avaliação do processo é um motivador para os desenvolvedores. Se a equipe considera que os objetivos são inalcançáveis (GOLDENSON E HERBSLEB, 1995), pode nem sequer tentar alcançá-los. Por outro lado, El-Eman et al. (1999) apontam que o estabelecimento de recomendações ambiciosas pode favorecer a MPS.

O grupo **"apoio e troca de conhecimento**" (Linha 3, Figura 20) refere-se (i) à necessidade e às vantagens do apoio técnico sobre como seguir o processo de desenvolvimento de software e (ii) aos benefícios do intercâmbio de conhecimentos*,* em particular do *compartilhamento de experiências,* para a aderência dos projetos ao processo de desenvolvimento de software. O principal objetivo tanto do apoio técnico quanto da troca de conhecimento é *aumentar o nível de compreensão* do processo de desenvolvimento de software, para que assim ele seja seguido de maneira mais fidedigna. Quanto mais compreensível melhor.

Promover *assistência* (GOLDENSON e HERBSLEB, 1995) pode ser uma boa prática de apoio ao uso do processo. Essa assistência pode acontecer de várias formas. A *troca de conhecimento* (HALL et al., 2007) também ajuda os desenvolvedores a aprender com casos reais e evita que conhecimentos importantes fiquem restritos a apenas alguns especialistas. O processo

de aquisição, documentação e distribuição de conhecimento ajuda os colaboradores a perceber os pontos fracos e entender por que os esforços da MPS são úteis (STELZER e MELLIS, 1999).

Segundo Hall et al. (2007), uma das mais importantes habilidades de um bom desenvolvedor, depois apenas de ser proativo, é compartilhar o seu conhecimento com a equipe. Conhecer *quem sabe o quê* pode intensificar a troca de conhecimento e facilitar o apoio ao uso do processo. Conhecer as habilidades e o nível de conhecimento dos profissionais pode orientar os planos de treinamento e melhorar a alocação de recursos, especialmente a atribuição de atividades de transferência de conhecimento. Um caso particular de intercâmbio de conhecimentos refere-se à *troca de conhecimentos entre as unidades* de desenvolvimento de uma organização, fator importante para o desenvolvimento distribuído de software. Quanto maior o intercâmbio de conhecimento entre as unidades, maior a redução das diferenças entre elas. Segundo Stelzer e Mellis (1999) uma estreita cooperação entre as unidades provê laços de reforço natural, melhora a compreensão e o conhecimento dos colaboradores, incentiva as pessoas a explorar sinergias e, conseqüentemente, melhora a produtividade e a qualidade.

Em particular, a *agilidade para atender aos pedidos de apoio* ao uso do processo influencia positivamente na aderência ao processo. Se a resposta é demorada um desenvolvedor pode desistir de executar um procedimento do processo ou executá-lo sem realmente entender o procedimento. Em ambos os casos a aderência será afetada. Fatores de outras categorias estão relacionados com a agilidade para responder aos pedidos de apoio ao uso do processo, como: *colaboração, disponibilidade de recursos humanos* para atuar na MPS, *conhecimento e experiência do pessoal* e *política*.

Ainda relacionados à categoria "ADERÊNCIA", e em particular à capacidade de usar o processo, merece destaque o grupo de fatores relacionados à **disponibilidade e adequação das ferramentas** de apoio ao desenvolvimento do software (Linha 4, Figura 20). Se as ferramentas não são adequadas (ou não estão disponíveis), o desempenho do processo pode ser diminuído. Sintomas de ferramentas inadequadas ou indisponíveis são: (i) inconsistência entre as ferramentas; e (ii) necessidade de trabalho adicional, tanto por dificuldades de *integração entre ferramentas* quanto porque atividades manuais podem ser necessárias (*automação* é um fator chave de sucesso para a MPS). A automação de processos relacionados ao desenvolvimento de software pode aumentar o desempenho do processo. No entanto, um cuidado especial que precisa ser considerado é o fator *automação de processos não bem definidos* (NASIR et al., 2008), ou seja, um procedimento pode estar automatizado, mas não ter sido bem definido antes da automatização e nesse caso, não haverá o ganho esperado para o desempenho do processo de desenvolvimento de software.

A aderência a um processo de desenvolvimento de software está relacionada à complexidade das suas **"normas e procedimentos"** (Linha 5, Figura 20). Quanto mais normas e procedimentos de um processo são complexos, mais difícil será executá-lo fielmente. Alguns desenvolvedores reportam, por exemplo, problemas com procedimentos de coleta de dados (BEECHAM et al., 2003). Um dos aspectos relacionados a normas e procedimentos é o fator *documentação*. Segundo Nasir et al.(2008), documentação é essencial para comprovações e disseminação formal de programas de MPS. Essa visão pode levar a uma barreira apontada pelos próprios autores, que seria a excessiva documentação e formalização. Beecham et al. (2003) apontam também para a dificuldade em coordenar e gerenciar a documentação. Uma conseqüência do excessivo foco em documentação observada em campo no SERPRO, além da burocratização do processo, é algumas vezes a execução do processo pelo processo, e desviando o foco que deveria ser em desempenho do processo e qualidade do produto final. Normas e procedimentos claramente estabelecidos, por outro lado, podem melhorar a padronização do processo e, como conseqüência, a reutilização de boas práticas e a aderência ao processo.

Por fim, um caso particular de influência negativa para aderência é o fator **pressão do tempo** (Linha 6, Figura 20), mencionado por Niazi et al. (2005a). Às vezes, as equipes de desenvolvimento deliberadamente abandonam alguns procedimentos do processo a fim de atender as restrições de tempo (BADDOO et al., 2007), mesmo quando isso pode significar a diminuição da qualidade do produto final. Adicionalmente, a pressão por tempo pode provocar a alocação de recursos para trabalhos que não estão associados a sua capacitação (BADDOO et al., 2007), causando, inclusive *divergência entre o treinamento obtido e as atividades executadas*.

3.3.4 Categoria Participação e Motivação

O grau com que os colaboradores participam das atividades de MPS foi citado como um fator chave de sucesso por quase toda a literatura encontrada. Segundo Nasir et al. (2008), a participação foi o principal fator chave para influenciar a MPS. Além disso, El-Eman et al. (1999) encontraram, com base em critérios com significância estatística, a relevância da participação do pessoal técnico na MPS. Segundo Dyba (2003), esse fator é particularmente importante para o sucesso das iniciativas de MPS em pequenas organizações.

Influenciando "ADERÊNCIA" e "MELHORIA CONTÍNUA", os fatores relacionados especificamente a "PARTICIPAÇÃO E MOTIVAÇÃO" (Figura 21) estão fortemente relacionados a fatores como: *apoio e troca de conhecimento*, *grau com que os empregados contribuem para a MPS* e *conhecimento e experiência do pessoal*.

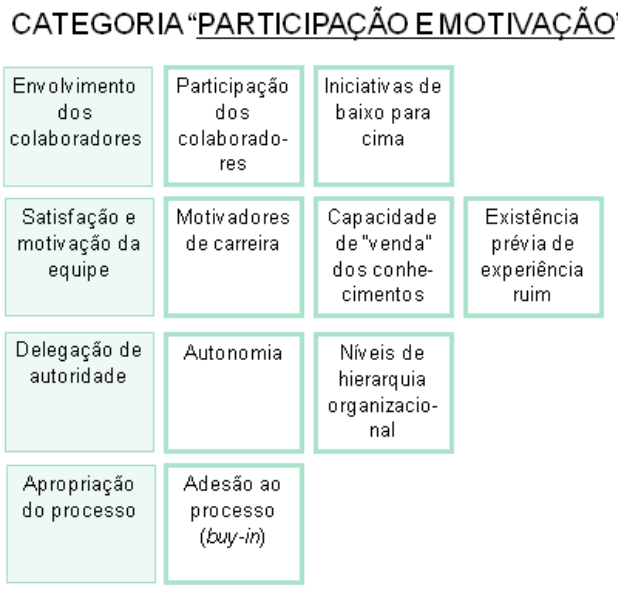


Figura 21: Fatores da categoria "Participação e motivação"

Os desenvolvedores são aqueles que vão experimentar os ganhos de um processo melhorado e, principalmente, vão ficar com o ônus se o processo funciona mal. Além disso, tais profissionais podem obter satisfação no trabalho ao produzir um processo de boa qualidade e produtos de alta qualidade (BADDOO e HALL, 2002). Assim, seria natural esperar um envolvimento elevado dos colaboradores na MPS (incluindo a participação técnica).

No entanto, esta participação nem sempre acontece como o esperado. Se os desenvolvedores sentem que as suas propostas não são consideradas ou não são tratadas em tempo, eles tendem a diminuir a contribuição. Ver as suas contribuições convertidas em partes do processo faz com que os desenvolvedores sintam as suas contribuições como significativas. O sentimento de **apropriação do processo** motiva os profissionais a contribuir e eles sentem-se co-responsáveis e co-autores do processo de desenvolvimento de software melhorado.

Adicionalmente, *cultura e existência prévia de experiência ruim* (também apontado na categoria "COMUNICAÇÃO") podem afetar a maneira como os atores entendem o processo e o executam. Por exemplo, se um processo de desenvolvimento de software baseia-se na delegação de autoridade e descentralização de poderes, mas é utilizado por desenvolvedores que preferem trabalhar com as tarefas determinadas, haverá uma lacuna entre o que o processo recomenda e o que é realmente executado.

A crescente *participação dos colaboradores* nos esforços da MPS pode trazer muitos benefícios (DYBA, 2001a; NIAZI et al. 2005) para o desempenho do processo e, assim, para a consecução dos objetivos do negócio. Empresas que adotaram mecanismos para aumentar a participação dos desenvolvedores em MPS, com base nesta crença, obtiveram resultados

positivos. Do ponto de vista organizacional, aumentar a oportunidade para contribuir pode, por si só, aumentar o envolvimento dos desenvolvedores em MPS. Alguns estudos apontam para a valorização de *iniciativas de baixo para cima* como fator que pode aumentar a participação da equipe e motivação.Quanto mais o poder é descentralizado (BADDOO e HALL, 2002) ou os funcionários têm *autonomia*, mais a MPS tende a ter sucesso. Em contrapartida quanto mais *níveis de hierarquia organizacional* existem, menos motivada é a equipe.

Alguns desenvolvedores vêem a MPS como uma oportunidade para reforçar a sua carreira. Além disso, as pessoas acreditam que a sua participação na MPS pode aumentar a *capacidade de venda dos seus conhecimentos*. Segundo Baddoo e Hall (2002), "*motivadores de carreira*" pode ser um bom argumento a ser usado pelo pessoal da MPS ao "vender" idéias de MPS para a gerência sênior.

Para entender melhor a relação entre os fatores desta categoria e a sua influência para a MPS, um diagrama de causa e efeito foi elaborado, considerando os principais componentes de um processo: pessoas, métodos, ferramentas e organização (Figura 22). Este diagrama destrincha os fatores apresentados na Figura 21 e, quando pertinente, refere-se a fatores apontados em outras categorias, que estão estreitamente relacionados à "PARTICIPAÇÃO E MOTIVAÇÃO".

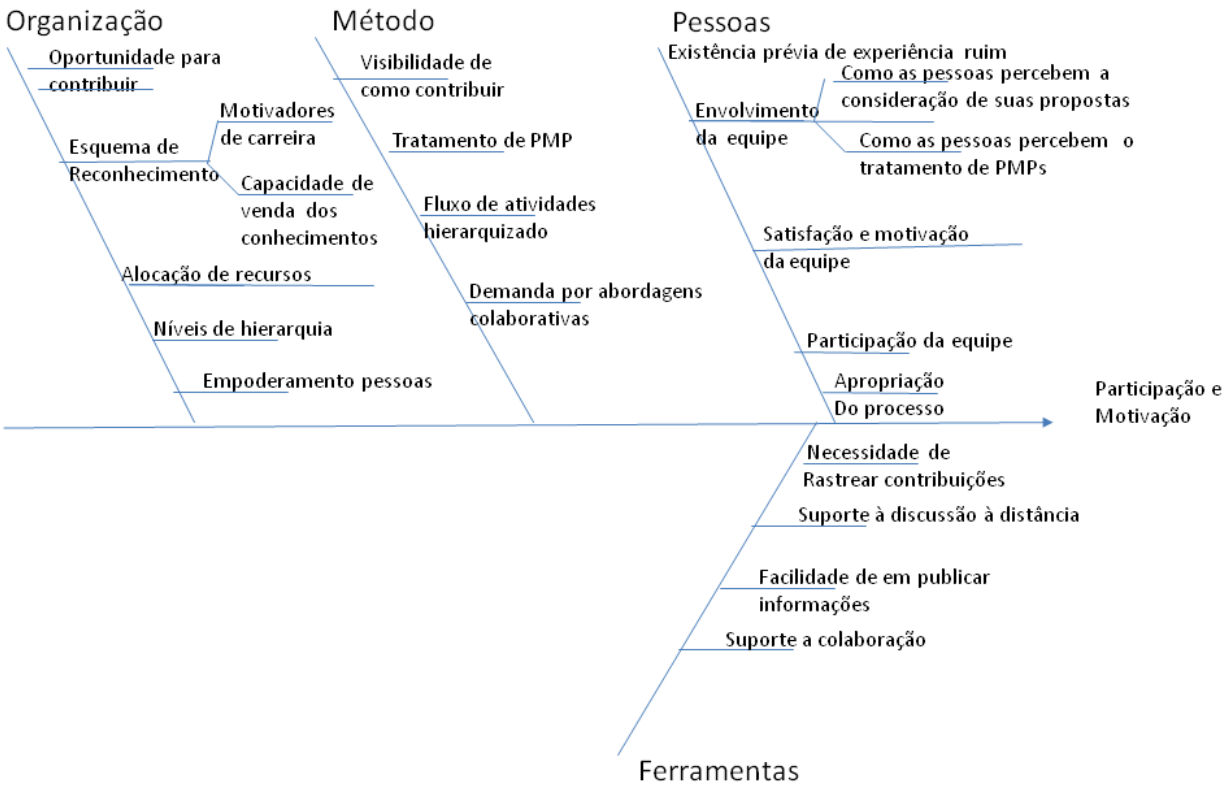


Figura 22: Diagrama de causa e efeito para "Participação e motivação"

Iniciativas bem-sucedidas de MPS enfatizam a importância da colaboração que, para o desenvolvimento distribuído de software, inclui a colaboração de funcionários de diferentes unidades distribuídas. A motivação para cooperar está relacionada a aspectos organizacionais, tais como estrutura organizacional, preocupação com a medição e orientação empresarial.

3.3.5 Categoria Comunicação

O grau em que os esforços de comunicação precedem e acompanham o programa de MPS e o grau em que os membros das diferentes equipes e diferentes departamentos se comunicam influenciam sobretudo a aderência, mas também a evolução do processo de desenvolvimento de software e a participação dos desenvolvedores na MPS. A ausência de instrumentos de apoio à colaboração e discussões, ou a falta de uma maneira fácil de rastrear contribuições e publicar informações (comunicação) podem influenciar negativamente a participação dos desenvolvedores. Quanto mais fácil for participar e observar o desdobramento das suas solicitações, maior a tendência das pessoas contribuírem para o processo.

Um forte esforço de comunicação deve preceder e acompanhar a implementação formal de um programa de MPS. Segundo Baddoo e Hall (2002), a relação entre comunicação e melhoria é forte, particularmente em grupos de desenvolvedores e de gerentes sênior. Mesmo unidades de desenvolvimento que detém alto grau de maturidade possuem problemas de comunicação que parecem impactar significativamente o desempenho dos projetos (HALL et al. 2007). Ademais Hall et al. (2007) sugerem que a operacionalização da comunicação no nível de processo, pode oferecer uma arma poderosa aos gerentes para melhorar os resultados de atividades da engenharia de software. Fatores diretamente relacionados à "COMUNICAÇÃO" são apresentados na Figura 23.

Figura 23: Fatores da categoria "Comunicação"

O incentivo a comunicação e colaboração é apontado como importante em 64% dos casos de organizações que implementaram a ISO e em 74% dos casos que implementaram o CMM, de acordo com observações de Stelzer e Mellis (1999). Niazi et al. (2005a) também citam este fator como chave para o sucesso da MPS. Dada a sua importância, o grupo comunicação foi detalhado em um novo diagrama causa e efeito (Figura 24). Este diagrama recorreu ainda à experiência do SERPRO, para detalhamento e complementação de fatores identificados relacionados à

comunicação. Por estarem muitas vezes associados (STELZER E MELLIS, 1999 e NIAZI et al., 2005a), fatores relacionados à colaboração também são mencionados na Figura 24.

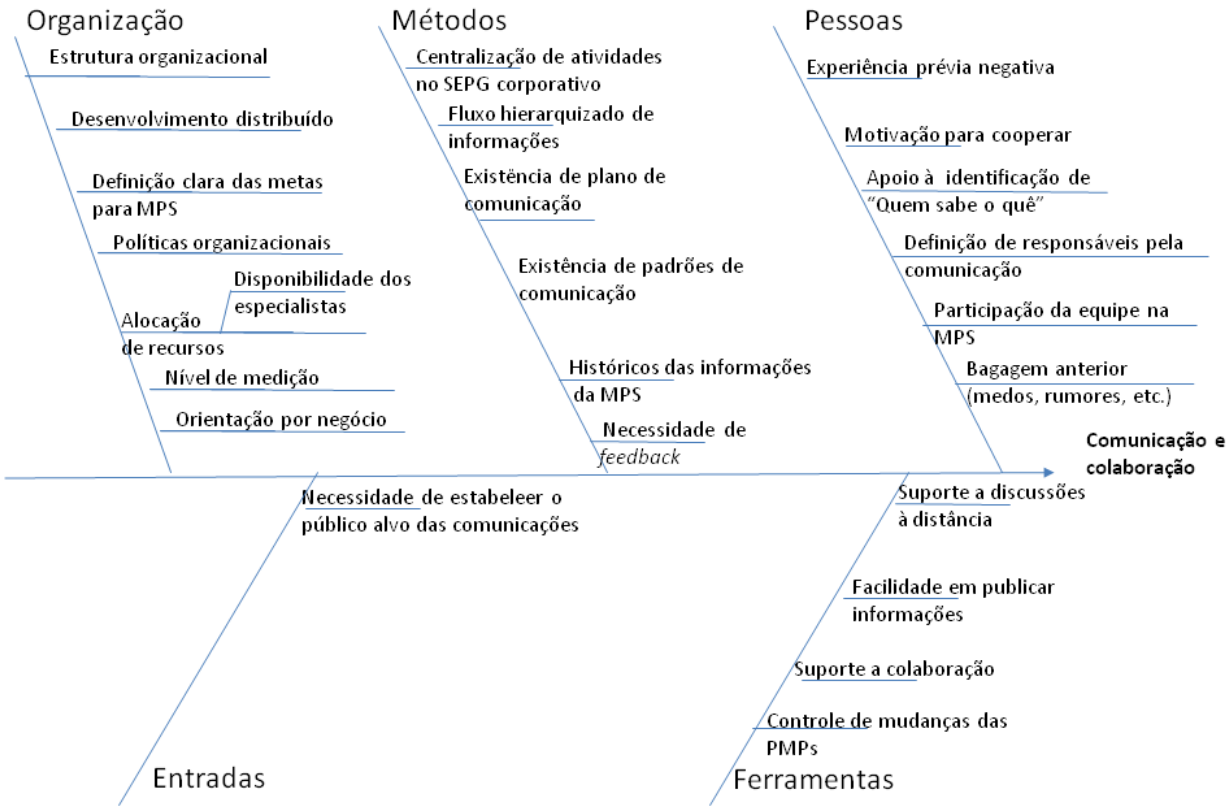


Figura 24: Analisando influências para comunicação e colaboração

Antes de empreender um esforço de MPS e tentar executar um processo, o *estabelecimento de metas claras para a MPS* é importante. Esses objetivos devem ser divulgados a todos os interessados. Ao entender o objetivo da MPS (EL-EMAM et al., 2001) é possível compreender o significado de cada atividade e também contribuir para a MPS.

A comunicação intensiva contribui para suprimir os boatos, afastar mal-entendidos, vencer a resistência, e dissipar os receios dos engenheiros de software (STELZER e MELLIS, 1999). Eles são sensíveis à *política* organizacional (fator apontado na categoria "ASPECTOS ORGANIZACIONAIS, Figura 18) e ao fantasma de *experiências ruins* no passado.

A comunicação entre membros da equipe pode ser alcançada por *Fóruns de MPS*, por exemplo. O *feedback* também é um mecanismo importante de comunicação que pode melhorar a aderência. Esse *feedback* deve acontecer em duas direções: SEPG para equipe de desenvolvimento e equipe de desenvolvimento para o SEPG. O *feedback* constante deve auxiliar a responder às questões: (i) Estou executando o processo com fidelidade? (ii) Eu estou alcançando as metas de MPS? E (iii) estou definindo um processo executável? (iv) Será que este processo pode ajudar a equipe a atingir seu objetivo?

Um caso particular do problema de comunicação, para organizações que trabalham com o desenvolvimento distribuído de software, é a falta de comunicação entre o SEPG e as unidades de desenvolvimento. A partir da observação em campo (Capítulo 4), perceberam-se poucos padrões de comunicação das metas de MPS e alguma dificuldade para publicar e manter as informações acerca do histórico e da gestão da MPS. Isso pode afetar a aderência ao processo na medida em que as unidades, por vezes, não entendem porque elas deveriam executar algumas atividades ou não se mantêm atualizadas. No sentido oposto, o SEPG pode não compreender as necessidades das unidades de desenvolvimento e, assim, definir um processo de difícil execução e que impacte o dia-a-dia local negativamente. Outro risco para uma MPS que envolva equipes distribuídas é o SEPG agir como um *centralizador* e não como um catalisador da MPS. Se existe *rigidez hierárquica* no fluxo de trabalho, exigindo a intervenção do SEPG corporativo em muitas aprovações de PMPs e na análise de dúvidas quanto ao uso do processo, a comunicação direta e a colaboração entre os profissionais intra ou extra-unidade podem ser inibidas.

Intensa comunicação ajuda a criar uma cultura organizacional coerente que é necessária para realizar melhorias substanciais. Mecanismos que ajudem a promover a comunicação podem ser muito úteis para uma MPS distribuída e colaborativa.

3.4 MPS e o mundo do software livre

Com a tendência de incorporar o software livre, ou o paradigma de desenvolvimento de software livre nas organizações, os fatores relacionados à melhoria de processos de desenvolvimento de software livre também precisam ser considerados. Adicionalmente, a definição das estratégias para MPS colaborativa e distribuída levou em consideração práticas do desenvolvimento de SL, por isso recomendações para processos de desenvolvimento de SL podem ser importantes.

No contexto do OMM (Seção 2.5.3), foram identificados elementos de confiabilidade do processo de software por meio de um questionário aplicado às empresas que utilizam, ou pretendem utilizar, software livre no seu desenvolvimento (QUALIPSO, 2008). A partir desses elementos e da experiência em MPS, a autora desta tese estruturou recomendações para processos de desenvolvimento de software livre. Também foi considerado o trabalho de Michlmayr (2005) que aponta características de processo de software livre que estão mais relacionadas a projetos de SL que obtiveram sucesso

Cada recomendação foi estruturada em: (i) identificação; (ii) objetivo principal; (iii) descrição e justificativa da importância da recomendação; (iv) ações recomendadas que possam

contribuir para alcançar o objetivo principal da recomendação; e (v) relacionamento da recomendação com os elementos de confiabilidade originais, quando foi o caso. Um exemplo de recomendação está disponível na Figura 25.

R1 – Participação da Comunidade

Monitorar o envolvimento da comunidade e evidenciar a atividade do projeto de SL. Depois que o projeto está em execução, a comunidade interagirá de diferentes maneiras com ele. A participação deve ser monitorada para assegurar níveis adequados de popularidade e apoio para a comunidade, bem como manter o projeto em movimento. Indicadores típicos da atividade de projeto e do envolvimento da comunidade são: a evolução dos downloads; evolução das inscrições nas listas de correio eletrônico, evolução do número de contribuintes e número de confirmações em um determinado período. Quanto mais as pessoas percebem a interação da comunidade, mais popular o projeto será. Promover a comunicação dentro da comunidade pode favorecer a sua participação.

- *Revisar periodicamente o envolvimento da comunidade e da atividade do projeto para assegurar que as interações estão ocorrendo. Exemplos de evidências de interação da comunidade: histórico de sugestões de novas funcionalidades, relatórios de erros, pedidos de apoio e suas respostas, listas de tarefas, etc.*
- *Promover o aumento de usuários e empenho dos usuários para a comunidade.*
- *Monitorar se o nível de comprometimento com as tarefas do projecto cresce continuamente e incentivar os usuários a, progressivamente, se envolver no processo de desenvolvimento do SL. Por exemplo, monitorar o progresso de desenvolvedores ativos transformando-se em desenvolvedores "commiters".*
- *Incentivar a comunidade a contribuir com código-fonte, documentação, sugestões, relatórios de bugs, tradução, etc.*
- *Monitorar a cobertura do projeto na mídia (blogs, revistas, fóruns, meios de comunicação de massa).*
- *Estabelecer um sistema de recompensa relacionado com as contribuições e sua utilidade. Por exemplo, promover desenvolvedores com base no seu índice de respostas úteis e contribuições pode ser uma boa prática.*
- *Seguir uma estratégia bem definida para atrair novos usuários.*

Elemento de confiança relacionados: Popularidade do produto (REP), Contribuição de Linhas de Produtos de Empresas de Software (CONT), Relações entre as partes interessadas (usuários, desenvolvedores, etc) (STK). from Software Companies (CONT), Relationship between stakeholders (users, developers etc) (STK).

Figura 25: Exemplo de recomendação para processos de desenvolvimento de SL

Tais recomendações foram organizadas em um relatório técnico (MALHEIROS et al., 2009a) e, posteriormente incorporadas ao documento de trabalho do OMM (WITTMANN et al., 2009). Foram estruturadas onze recomendações: comunicação; uso de padrões conhecidos de SL; documentação do projeto; ambiente integrado de desenvolvimento; qualidade do produto de SL; melhoria contínua do projeto; gestão de configuração; gestão de recursos; processo de desenvolvimento; licença; e avaliação do produto e do projeto.

A estruturação dessas recomendações foi um passo intermediário para estabelecer as práticas do OMM e para definir a estrutura final do modelo.

3.5 Considerações finais

Vários estudos têm analisado as iniciativas de MPS para identificar as influências sobre o seu sucesso (ou insucesso). Muitos fatores (motivadores ou obstáculos) foram relatados em

trabalhos primários e secundários na literatura. Organizar, descrever a lógica das relações e analisar tais fatores que podem influenciar a MPS foram passos importantes para confirmar que:

- Fatores para a MPS identificados a partir da experiência do SERPRO são comparáveis aos motivadores e barreiras identificados na literatura. Assim, as lições aprendidas com a experiência do SERPRO podem ser úteis para outras empresas e contextos; e
- Alguns fatores que influenciam a MPS podem ser comparados aos fatores que influenciam o desenvolvimento de software, e em particular o DDS. Como exemplo, descobriram-se fatores relacionados à gestão do conhecimento, motivação pessoal, colaboração e problemas de comunicação. Essas observações estão de acordo com o entendimento de que a MPS pode beneficiar-se de práticas que têm sido adotadas para desenvolvimento de software.

Apesar dos vários relatos de experiência em MPS em diferentes níveis de detalhamento; a literatura carece de trabalhos que estabeleçam estratégias para a MPS (NIAZI et al., 2005a). Ou ainda, que reportem em detalhes as estratégias que estão sendo utilizadas.

Após a organização dos fatores, foi possível confirmar a intuição de que boa parte dos fatores da MPS está relacionada de alguma forma com a comunicação, participação da equipe, a natureza da própria evolução e implementação de planos (estratégias) definidos para melhorar o processo. Muitos desses fatores podem ser abordados e beneficiar-se de estratégias colaborativas e distribuídas de MPS e suas concretizações em uma infraestrutura de apoio.

Apesar de tais esquemas não terem sido estruturados com base em dados quantitativos de co-relação entre os fatores, o que é uma limitação, os esquemas de fatores da MPS apresentados podem ser úteis para a definição de uma estratégia colaborativa e distribuída de MPS e para direcionar a definição de critérios para avaliar as iniciativas de MPS, uma vez que alguns grupos de fatores identificados podem ser convertidos em objetivos e desdobrados em indicadores, seguindo o paradigma GQM - *Goal Question and Metric* (BASILI et al., 1994a). Agrupar os fatores em categorias relacionadas pode facilitar a identificação de mecanismos para tratar um conjunto de influências.

Compreender os fatores que influenciam programas de MPS e as recomendações para processos livres de software foram importantes também para subsidiar a definição de estratégia de evolução do OMM (Capítulo 6, Seção 6.3).

As lições aprendidas com as observações em campo e com a vivência da autora em um programa de MPS, que auxiliaram na organização deste capítulo, são reportadas no próximo capítulo.

4- Experiências em MPS e GC aplicadas ao desenvolvimento de software no SERPRO

Neste capítulo, após algumas considerações iniciais (Seção 4.1), são apresentadas a experiência em MPS no SERPRO (Seção 4.2) e observações do desenvolvimento de software no SERPRO sob a lente da GC (Seção 4.3).

4.1 Considerações iniciais

Modelos de referência em MPS (Capítulo 2) vêm sendo utilizados, nos últimos anos, pelas empresas para estruturar seus processos. Esses modelos reúnem boas práticas de engenharia de software e podem guiar a melhoria da capacidade dos processos de desenvolvimento de software.

Entretanto, a utilização desses modelos na prática requer adaptação. A MPS está sujeita a vários fatores críticos de sucesso não-técnicos, como necessidade de: participação dos desenvolvedores na MPS, comunicação ou troca de conhecimento (Capítulo 3). A influência desses e de outros fatores requer interpretação do modelo, seu desdobramento em processos e estratégias de aplicação, e sua adequação, para que eles sejam aplicados em casos reais.

Essa característica reforça a importância de estudos de caso, relatos de experiência e outros estudos experimentais como insumo para evoluir as pesquisas na área. Shull et al. (2001) afirmam que usar estudos experimentais para analisar o desenvolvimento de software sob condições reais pode auxiliar a validação de tecnologias maduras e a identificação de problemas em tecnologias menos maduras. Essa afirmação pode ser estendida também para a MPS.

Assim, em complemento à revisão da literatura apresentada no Capítulo 3, foram mapeadas as experiências em MPS no SERPRO, que possui um programa de melhoria contínua do desenvolvimento de software, desde 2000. A autora desta tese participa desse programa, desde 2001, em diferentes papéis. Lições aprendidas com o programa de MPS no SERPRO foram consolidadas e práticas do desenvolvimento de software foram analisadas à luz da gestão

do conhecimento. Com base em tal experiência, apresentam-se as lições extraídas da pesquisa que apoiaram a definição da estratégia ColabSPI proposta no Capítulo 5.

Vale ressaltar que as observações em campo são referentes ao período de 2005 a 2008 e que, no primeiro semestre de 2009, a empresa passou por uma reorganização das suas equipes de desenvolvimento de software, além de ter adaptado algumas soluções mencionadas, reflexo de alterações na alta direção da empresa.

Boa parte das informações apresentadas neste capítulo foram, primeiramente, registradas nos trabalhos: Malheiros et al. (2006); Malheiros et al. (2008a) e Malheiros et al. (2007b).

4.2 Experiência em MPS no SERPRO

Na época dos estudos experimentais aqui apresentados, o SERPRO era dividido em Unidades de Gestão (UG). Cada UG era gerenciada por uma superintendência e possuía uma ou mais projeções espalhadas em regionais ou escritórios. As UG estavam organizadas por área de atuação. Existiam duas classes principais de área de atuação: áreas de negócio e áreas de suporte. As áreas de negócio faziam atendimento e desenvolvimento de software para cada cliente e estavam divididas de acordo com o negócio do cliente, por exemplo, área de negócio Administração Tributária (SUNAT) ou área de negócio Comércio Exterior (SUNCE). Para a MPS, as UG de negócio eram as mais relevantes, uma vez que eram elas que possuíam as equipes de desenvolvimento de software.

As equipes de desenvolvimento de software estavam, e continuam, distribuídas geograficamente e precisam desenvolver sistemas de forma integrada (especialmente com as demais unidades de desenvolvimento que trabalham numa mesma linha de negócio). As projeções regionais de desenvolvimento eram denominadas de pólos de desenvolvimento e eram administradas por um gerente e um grupo de líderes de projeto. Durante o período observado, o número de pólos foi alterado algumas vezes e variou de 20 a 40 pólos de desenvolvimento em todo Brasil. Cada UG era supervisionada por uma das diretorias. O Programa de Melhoria de Processo do SERPRO (Seção 4.2.2) foi vinculado à diretoria, em nível estratégico. Informações atualizadas da estrutura da empresa podem ser obtidas na página institucional[45].

4.2.1 Processo SERPRO de Desenvolvimento de Soluções (PSDS)

Os sistemas desenvolvidos pelo SERPRO utilizam diversas soluções tecnológicas e apresentam variações de plataforma, sistemas operacionais, linguagens de programação (ex.:

[45] http://www.SERPRO.gov.br/instituicao/estrutura/organograma

Java, Visual Basic, C, Natural, COBOL, Python) e paradigmas de desenvolvimento (ex.: orientado a objeto, estruturado). Ao mesmo tempo, a abrangência em termos de área de atuação, a dispersão geográfica e a quantidade de pessoas e equipes de desenvolvimento potencializam a complexidade da gestão do desenvolvimento e da melhoria do seu processo.

Por isso, o SERPRO criou em 2001 um processo padrão para desenvolvimento e manutenção de soluções (PSDS), primeiro resultado de um programa corporativo para a modernização do desenvolvimento, criado em 2000.

O objetivo do PSDS é fornecer um processo de software padronizado e dinâmico às suas equipes de desenvolvimento, que consolide as melhores práticas de desenvolvimento de software. O PSDS oferece a base para responder às perguntas: "quem faz o quê", "quando" e "como", no contexto de desenvolvimento e manutenção de soluções. Sua estrutura está baseada em macroatividades (disciplinas), um conjunto relacionado de atividades que podem ser referentes a: Negócio, Gestão de Processo, Gestão de Projetos, Engenharia ou Suporte. Todas as macroatividades foram adaptadas às melhores práticas definidas nos modelos CMM e CMMI.

O PSDS está em constante evolução. À medida que vai se tornando mais maduro, novas macroatividades são inseridas, atualizadas ou removidas. A última versão disponível durante o período avaliado, a 6.7, contemplava as seguintes macroatividades: (i) Negócio; (ii) Requisitos; (iii) Análise; (iv) Projeto; (v) Implementação; (vi) Teste; (vii) Homologação; (viii) Implantação; (ix) Gestão de Projetos de Software; (x) Gestão de Configuração de Software; (xi) Medição e Análise; (xii) Garantia da Qualidade de Software; (xiii) Revisão por Par; (xiv) Gestão de Aquisição com o Fornecedor; (xv) Programa de Treinamento; (xvi) Gestão do Processo da Organização; e (xvii) Análise e Decisão.

O *site* do PSDS foi inspirado em elementos do RUP (IBM, 2010), seguindo inclusive alguns de seus padrões de modelagem. O RUP é um arcabouço de processos de engenharia de software, de propriedade da IBM, que oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento e foi baseado no processo unificado (JACOBSON et al., 1999). A macroatividade de Gestão de Projetos utilizou também como referência o PMBoK (*Project Management Body of Knowledge*) (PROJECT MANAGEMENT INSTITUTE (PMI), 2004).

Materializado como um portal web (Figura 26), o PSDS está disponível para todos os funcionários da empresa. Sua tela principal disponibiliza uma árvore à esquerda composta de todas as macroatividades. Os componentes de cada macroatividade são: (i) definição; (ii) objetivos; (iii) atividades; (iv) artefatos; (v) fluxo de atividades; e (vi) visão geral.

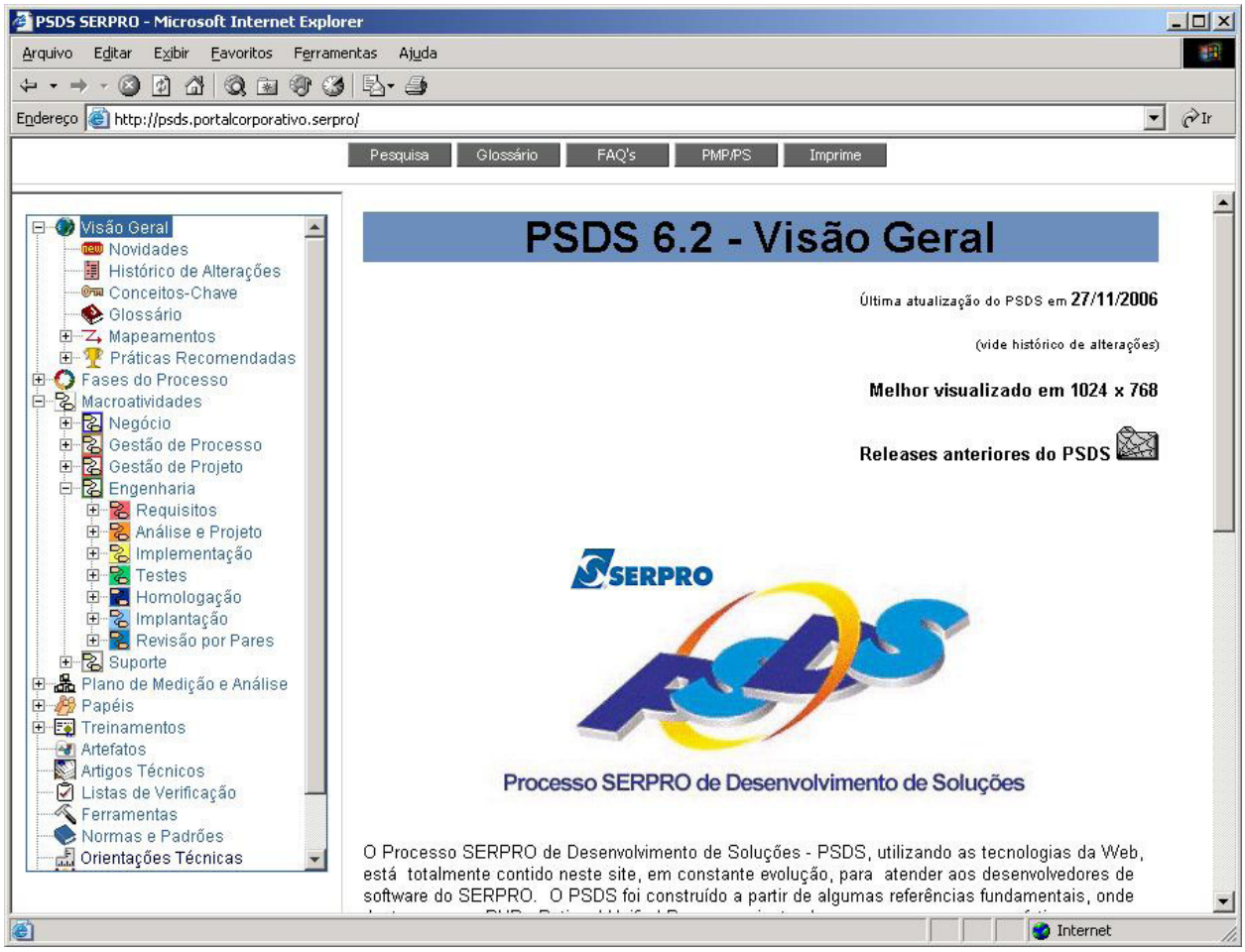


Figura 26: Tela principal do PSDS (SERPRO, 2007)

Para cada atividade, são disponibilizadas: (i) descrição; (ii) responsáveis; (iii) participantes; (iv) entradas; (v) saídas; (vi) critérios de entrada; (vii) critérios de saída; (viii) recomendações; e (ix) orientações técnicas. Existem modelos de artefatos ou indicação de ferramenta específica disponíveis para as entradas e saídas de cada atividade do processo.

O PSDS contém procedimentos definidos, inclusive, para incorporação de melhorias ao próprio processo. Gestão do Processo da Organização (GPO) é a disciplina do PSDS que descreve os procedimentos para a definição, manutenção e adaptação do próprio processo de desenvolvimento de software. Ela também estabelece atividades de melhoria contínua da estrutura e conteúdo do PSDS, de forma que este esteja sempre atualizado e ajustado à realidade das diversas áreas de desenvolvimento da empresa. Essa disciplina organiza as práticas preconizadas no CMMI, nas áreas de processo OPF e OPD (Capítulo 2), para definir práticas corporativas próprias de gestão do PSDS. As atividades previstas para a GPO são:

- Avaliar o uso do processo de software;
- Gerenciar propostas de melhoria de processo;

- Planejar a evolução do processo;
- Evoluir o processo da organização;
- Controlar a geração de *releases*;
- Acompanhar a evolução do processo; e
- Prover suporte (apoio) ao processo.

Com alguma simplificação, essas atividades foram consideradas para estabelecer os passos para a MPS da ColabSPI (Capítulo 5).

4.2.2 Programa SERPRO de Melhoria do Desenvolvimento

Os objetivos do programa para modernização do desenvolvimento de software criado no SERPRO eram: (i) estabelecer políticas para o desenvolvimento de software; (ii) definir um processo padrão para todas as unidades de desenvolvimento de software da empresa; e (iii) implantá-lo em toda a organização. Desde o início, a melhoria contínua do desenvolvimento era uma preocupação e o CMM foi escolhido, na época, como modelo de referência.

O PSDS foi um dos principais resultados desse programa. Já em 2002, o SERPRO obteve o primeiro reconhecimento internacional, decorrente dos esforços empreendidos para melhorar o desenvolvimento: a certificação oficial da primeira unidade em CMM Nível 2. No ano seguinte, mais três unidades foram certificadas oficialmente. A próxima meta estabelecida para o programa foi nivelar o conhecimento obtido com essas unidades com as outras unidades distribuídas. Uma sistematização preliminar de estratégias para implantar o processo padrão em uma unidade de desenvolvimento distribuída (NASCIMENTO e MALHEIROS, 2004) foi elaborada e evoluída, culminando a posteriori, na Sistemática de Avaliação de Maturidade e SERPRO (SERPRO, 2006).

O programa de modernização passou por algumas reformulações para mantê-lo sempre alinhado aos objetivos estratégicos da empresa. Entre 2005 e 2008, o Programa SERPRO de Melhoria do Desenvolvimento de Soluções - PSMDS - esteve implementado na estrutura organizacional como uma Unidade de Alinhamento Estratégico e, em seguida, como um programa da Coordenação de Tecnologia, sendo, em ambos os casos, supervisionado por um diretor da empresa. O vínculo à diretoria foi um reflexo da importância dada à MPS na empresa. O objetivo do programa era, ao mesmo tempo, evoluir o PSDS e institucionalizá-lo nas unidades distribuídas de desenvolvimento.

Com a reestruturação da empresa, em 2009, a equipe do programa de MPS foi desmembrada e ampliada, ficando uma parte focada em inovação e questões estratégicas e outra

parte focada na institucionalização e nas questões tático-operacionais. As observações registradas nesta seção não consideram esta nova estrutura.

O programa de MPS visava a garantir treinamento, recursos e fundos para incorporação do PSDS pelas unidades de desenvolvimento (mais de 40, em determinado momento) e mantê-lo alinhado com as boas práticas de mercado (MALHEIROS et al. 2006) e com os objetivos estratégicos da empresa. O programa abordava duas frentes principais: (i) **evolução** contínua do processo de desenvolvimento e (ii) sua **institucionalização** (uso) em todas as unidades de desenvolvimento.

As diretrizes estabelecidas para esse programa foram (MALHEIROS et al. 2006):

- manter e evoluir o PSDS, tendo como referências os modelos CMM e CMMI;
- monitorar o uso do PSDS pelas unidades diretamente envolvidas com o desenvolvimento de soluções de software;
- zelar pela simplicidade e praticidade na definição dos procedimentos e padrões do PSDS, visando à melhoria da qualidade das soluções desenvolvidas;
- fortalecer grupos de qualidade de software como indutores do uso adequado do processo;
- buscar continuamente a melhoria no nível de atendimento aos clientes;
- definir requisitos de ferramentas de apoio ao processo; e
- buscar a integração com outros processos e projetos empresariais.

Na Figura 27, os papéis organizacionais definidos para viabilizar essas diretrizes são apresentados.

O grupo Coordenação do PSDS (**CPSDS**) foi criado para coordenar o programa de MPS. Esse programa continha dois projetos: o Projeto Estratégico de Melhoria de processo (**PSMPS**) - equivalente ao SEPG do CMMI; e o Projeto Garantia da Qualidade de Software Corporativa (**PSGQS**), grupo que coordenava os consultores de Garantia da Qualidade de Software (GQS) da empresa. Os três grupos faziam parte da Unidade de Alinhamento Estratégico (UAE) (Figura 27) e foram formalmente estabelecidos.

O PSMPS era o grupo corporativo que centralizava os esforços de MPS. Seu papel era garantir uma visão geral dos esforços investidos na MPS, agindo como facilitador e direcionador desses esforços, buscando colaboração e colaborando, com as partes interessadas em MPS.

Os demais grupos, representados na Figura 27, não estavam na estrutura organizacional formal da empresa e foram criados com base em designações específicas. O SERPRO optou por um modelo matricial (PROJECT MANAGEMENT INSTITUTE, 2004) para a MPS. À exceção

dos grupos da UAE, os grupos que trabalhavam com MPS eram formados por colaboradores das diversas áreas de desenvolvimento, que trabalhavam em tempo parcial para a MPS.

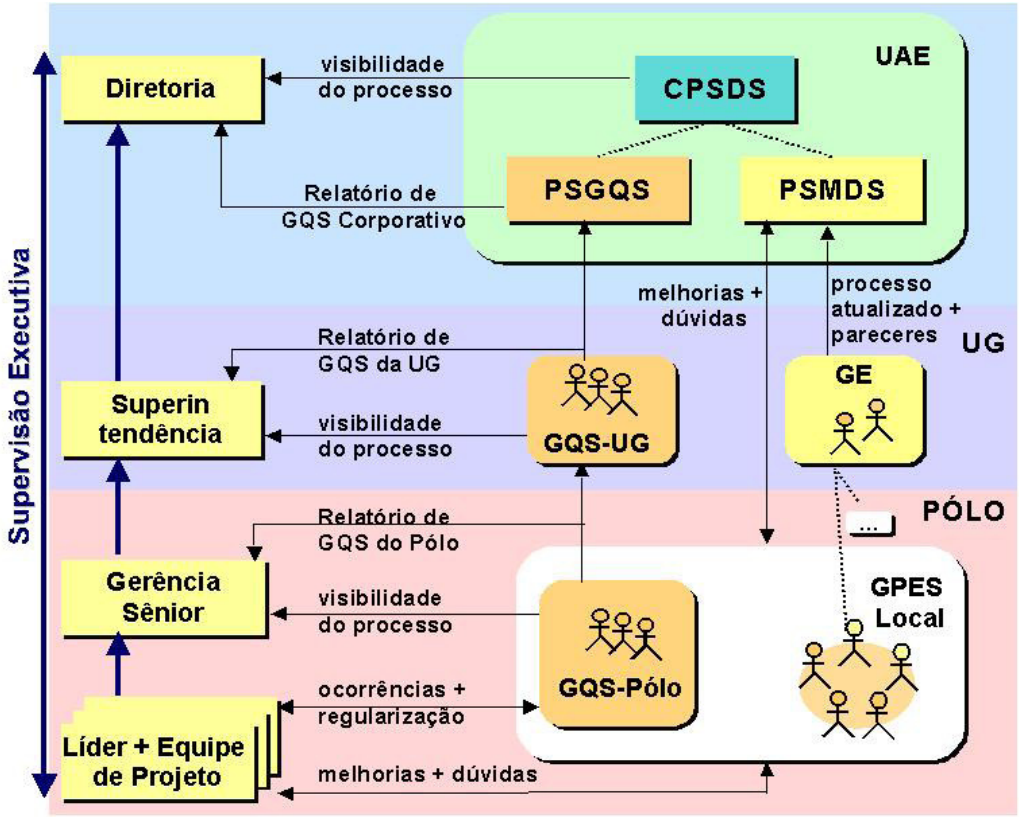


Figura 27: Estrutura da MPS no SERPRO (MALHEIROS et al., 2008a)

Esse modelo tem a vantagem de aumentar a participação dos desenvolvedores, grandes interessados na evolução e uso do processo. Em contrapartida, ele demanda uma estrutura organizacional de MPS que apóie a colaboração e, de alguma forma, garanta a motivação das pessoas. Como, muitas vezes, a prioridade dos desenvolvedores será o desenvolvimento de sistemas e não a MPS, aspectos motivacionais são fundamentais para o sucesso de um programa de MPS. No programa do SERPRO, a participação de um desenvolvedor em equipes da MPS foi considerada um fator positivo durante sua avaliação funcional.

Os Grupos Especialistas (**GE**) são os mantenedores dos ativos do processo. Na época do PSMPS, o SERPRO tinha doze grupos especialistas (ex.: Gestão de Projetos, Teste de Software, Requisitos). Cada grupo especialista estava focado em uma macroatividade e seu objetivo era apoiar o grupo de engenharia do processo na análise de propostas de melhoria de processo, estimando ganhos e perdas de cada mudança e o tempo necessário para implementá-las. Também era papel do GE realizar as alterações no processo que foram aprovadas pelo PSMPS. O foco do GE era a **evolução do processo**. Os participantes desses grupos tinham dedicação parcial às

atividades de MPS e atuavam prioritariamente como analistas de sistemas, projetistas, analistas de requisitos, líderes de projeto, testadores, entre outros papéis no desenvolvimento de software.

Os Grupos de Processo Locais (**GPES Local**) eram os guardiões do processo nas unidades. O foco principal desses grupos, também formados por desenvolvedores, era a **institucionalização do processo**. Suas atribuições eram: (i) apoiar os projetos de software no seu âmbito de atuação quanto à utilização do PSDS; (ii) garantir que o processo de desenvolvimento de software adotado no âmbito dos projetos tenha sido adaptado apropriadamente, a partir do PSDS, conforme diretrizes nele contidas; (iii) realizar avaliações da aplicação do PSDS em sua unidades, conforme periodicidade e procedimentos definidos no PSDS; (iv) identificar e incentivar a adoção de pontos fortes do processo; e (v) conduzir atividades de suporte de primeiro nível no seu âmbito de atuação. Cada unidade de desenvolvimento tinha um grupo de processo local. A estrutura de cada GPES Local era determinada pelo gerente da unidade de desenvolvimento. Algumas unidades optaram por montar grupos com dedicação parcial dos desenvolvedores, no mesmo modelo concebido para os Grupos Especialistas (GE). Outros gerentes preferiram alocar um coordenador de GPES Local dedicado exclusivamente a promover a institucionalização do PSDS na unidade. As diferentes abordagens levaram em consideração questões culturais e de perfil e disponibilidade dos recursos. O fato das estruturas internas do GPES Local serem diferentes não foi um problema para a implantação do PSDS e a padronização do desenvolvimento nas unidades. As vantagens e desvantagens de cada abordagem foram objeto de estudo. Além das atividades de institucionalização, o GPES local (Figura 27), colaborava com a evolução do processo identificando e submetendo propostas de melhoria de processo e realizando pré-analise das propostas de melhoria originadas no seu âmbito de atuação.

Os GQS eram os consultores de Garantia da Qualidade de Software, que verificavam a aderência dos projetos de software ao processo padrão nas unidades de desenvolvimento (pólos). Esses consultores identificavam problemas de aderência ao processo e geravam ocorrências a serem regularizadas pelas equipes de desenvolvimento. Cada pólo tinha a sua estrutura de GQS, coordenada por um GQS do pólo que, periodicamente, dava visibilidade do uso do processo para o gerente do pólo e para os GQS da UG. Os GQS da UG consolidavam as informações recebidas de todos os pólos por unidade de negócio e garantiam a visibilidade do processo no nível da UG.

A melhoria contínua do processo de desenvolvimento de software era feita por meio de Propostas de Melhoria de processo (**PMP**), que eram tratadas por meio de um fluxo de trabalho de análise que permeia o GPES Local, PSMPS e os GE. Os grupos GQS também contribuíam para a MPS ao identificar e submeter PMP para análise. À medida que o processo era utilizado

pelas unidades de desenvolvimento e implantado em novas unidades, oportunidades de melhoria eram identificadas e propostas pelos desenvolvedores.

A proposição e análise de PMPs eram controladas pela solução GM-PSDS (Gestão de Mudanças do PSDS), desenvolvida pela empresa. Todo o ciclo de vida de uma PMP desde sua geração até sua aprovação ou rejeição era acompanhado pela GM-PSDS. A contribuição da GM-PSDS para a MPS e, como pano de fundo, os processos organizacionais que lhes davam suporte, foram apresentados em Malheiros et al. (2006).

A partir do segundo semestre de 2007, houve uma reformulação na estrutura do SERPRO e as Unidades de Alinhamento Estratégico foram eliminadas. O assunto PSDS e sua melhoria passaram a ser tratados pela Coordenação de Tecnologia. Uma estrutura organizacional semelhante para a MPS foi criada, basicamente alterando os nomes dos projetos, mas mantendo sua finalidade e o seu nível estratégico de alinhamento com a diretoria.

Com a experiência em MPS no SERPRO (MALHEIROS et al. 2008), foi possível identificar algumas questões que precisam ser tratadas por uma organização que tenha várias unidades de desenvolvimento de software:

1) Como o processo padrão pode atender à necessidade de cada uma das unidades da organização? E como, ao mesmo tempo, garantir que mudanças no processo padrão são válidas para toda a organização?
2) Como evoluir o processo mantendo sua confiabilidade e integridade?
3) Como fazer com que o processo seja um agente motivador à troca e manutenção do conhecimento organizacional?
4) Como garantir que o processo padrão tenha informações suficientes para assegurar uma identidade mínima e a comunicação entre as unidades, inclusive favorecendo uma fácil adaptação no caso de troca de pessoas entre equipes?
5) Como garantir a disseminação do conhecimento acerca do processo para milhares de desenvolvedores com diferentes culturas e distribuídos em diferentes localidades?
6) Como avaliar o uso do processo nas unidades da organização (incluindo seu nível de maturidade)?
7) Como aumentar a possibilidade de propagar o sucesso obtido em uma unidade para as outras unidades?
8) Como oferecer condições/apoio para a MPS considerando variáveis como: (a) equipes geograficamente distribuídas; (b) custo para implantação de um programa de

MPS e avaliações oficiais; e (c) grande contingente de desenvolvedores com perfis muito diferenciados?

9) E, por fim, os usuários do processo estão trabalhando de uma forma melhor, principal objetivo da MPS (WIEGERS, 1998)?

4.2.3 Identificação de melhorias para o PSDS, para a MPS e as lições aprendidas

Lições aprendidas e fatores que influenciam a MPS no SERPRO foram identificados para subsidiar a definição da estratégia para a MPS. Fontes de informação diversificadas foram utilizadas: (i) relatórios de acompanhamento do uso do PSDS e de avaliação das unidades distribuídas de desenvolvimento de software, gerados a partir de entrevistas locais; (ii) PMP registradas na GM-PSDS; e (iii) reuniões da coordenação do PSDS e reuniões e relatórios do GPES Local.

Entre os anos de 2005 e 2009, foram realizadas várias visitas para acompanhar o uso do PSDS nas unidades distribuídas de desenvolvimento de software do SERPRO. No decorrer dos eventos de consultoria e de avaliação, equipes de desenvolvimento foram entrevistadas para avaliar a aderência dos projetos de software da unidade ao PSDS.

Mais de 30 visitas de consultoria e avaliação foram realizadas, sempre coordenadas por um dos dois membros do SEPG corporativo do SERPRO alocados para acompanhar o uso do processo e as iniciativas de MPS das unidades. As visitas foram programadas de acordo com sistemática de avaliação do SERPRO (SERPRO, 2006).

A partir das visitas de consultoria e das avaliações das unidades, pôde-se identificar problemas recorrentes em projetos. Práticas sistematicamente não cumpridas indicaram pontos do processo padrão que precisavam ser melhorados ou precisavam reforçar a capacitação. Adicionalmente, uma das etapas da sistemática de avaliação do SERPRO é a condução de entrevistas. A sistemática recomendava que gerentes seniores e líderes de projeto fossem entrevistados isoladamente, enquanto os desenvolvedores eram entrevistados em grupos. Técnicas de entrevistas foram utilizadas, como iniciar as entrevistas com perguntas mais fáceis.

Depois de algumas visitas, identificou-se que tais entrevistas eram também uma oportunidade para obter *feedback* direto para a melhoria do PSDS, já que algumas sugestões para o PSDS surgiam nas respostas relacionadas ao seu uso. A partir desse ponto, recomendações para a melhoria do PSDS passaram a ser identificadas de uma forma mais sistemática nas entrevistas, em complemento ao mecanismos de coleta de PMP. Duas perguntas abertas foram

incluídas nos roteiros de entrevistas: "*Cite algo pode ser melhorado no processo*" e "*Se você tivesse que mudar algo para melhorar qualidade, o que seria?*".

Recomendações foram sistematicamente coletadas nas entrevistas individuais ou em grupo em dez visitas a unidades distribuídas. No total, 230 pessoas foram entrevistadas. Entre elas: 10 gerentes seniores; 38 líderes de projeto e 182 desenvolvedores. Vale ressaltar que alguns desses desenvolvedores também desempenhavam papel para a MPS.

As sugestões de melhoria para o PSDS identificadas nas entrevistas (recomendações) variaram desde questões muito pontuais como "*Ajustar incompatibilidade entre o FAQ do PSDS e o manual ... quanto à reprogramação de prazo*" a recomendações mais gerais como "*Enxugar e/ou automatizar mais o processo de software, evitando redundância e excesso de documentação*". As sugestões foram registradas nos relatórios finais das visitas. Vale ressaltar que pontos fortes da MPS também foram registrados nos relatórios, mas esses pontos não serão reportados nesta seção. Desde sua coleta, muitas das recomendações já foram tratadas no PSDS. Ainda assim, elas são relevantes por apontar para fatores críticos de sucesso da MPS.

Cada sugestão identificada foi generalizada em um fator que a representasse. Dessa forma, alguns fatores de influência para a MPS no SERPRO começaram a ser identificados. Na identificação de fatores, recomendações muito específicas foram desconsideradas. Na Tabela 5, é apresentado um painel consolidado das principais recomendações de melhoria identificadas durante as entrevistas, já organizadas de acordo com os fatores aos quais elas deram origem. Ainda na Tabela 5, são indicadas as unidades que registraram a recomendação. A partir das reuniões da coordenação do PSDS e com os GPES locais, relatórios das unidades e de GQS, além das PMPs direcionadas para a disciplina de Gestão do processo da Organização do PSDS, foram identificados outros fatores que podem influenciar a MPS e são apresentados na Figura 28.

Por meio de um gráfico de causa e efeito foi possível estabelecer uma relação inicial entre os fatores para direcionar o plano de melhoria. Muitos fatores foram encontrados também na literatura revisada, à exceção de: "unidades em diferentes níveis de maturidade", "projetos com características diferentes, demandam processos diferentes", "diferentes clientes com necessidades diferentes", "baixo intercâmbio de conhecimento entre as unidades", "demora no atendimento de PMP", "troca da diretoria", "demora no esclarecimento de dúvidas" e "necessidade de integração entre ferramentas". Os itens em negrito (Figura 28) foram detalhados em novos gráficos causa e efeito.

Logo no início das observações percebeu-se que a lentidão no tratamento de PMP era um fator de diminuição da satisfação dos desenvolvedores e a credibilidade da MPS. As principais

etapas do tratamento de PMP (coleta, análise, desenvolvimento e entrega) e as possíveis barreiras à sua tempestividade foram identificadas para cada uma dessas etapas (Figura 29). Essas barreiras foram organizadas como um diagrama de causa e efeito e nortearam a definição dos mecanismos de apoio ao tratamento de PMPs (Capítulo 5). As causas em negrito são aquelas que podem ser mitigadas pelo uso de um mecanismo de apoio a documentação de processos (autoria e controle de versão) ou de um mecanismo de tratamento de PMP, como propostos na ColabSPI.

Tabela 5: Consolidado de recomendações ao PSDS coletadas em visitas às unidades

Fator	Recomendação para o PSDS	Equipes
Existência de Burocracia	Simplificar o processo de gestão para projetos sumários e resumidos, otimizando seu planejamento e acompanhamento	1,2, 9
	Enxugar o processo de software [PSDS], evitando redundância e excesso de documentação	1, 2,3, 6
	Otimização do processo [PSDS], objetivando a redução do custo de gestão para projetos pequenos.	7
Proporcionar melhor compreensão	Esclarecer melhor no PSDS os conceitos de...	6,1
	Explicitar nos modelos orientação para fazer ...	9
	Reforçar o conceito e a importância do planejamento e execução de revisões formais com o cliente	9
Lições aprendidas/ Troca de conhecimento	Implantar uma base histórica para consolidação dos resultados de projetos.	1
	Gerar exemplo de projeto sumário com informações mínimas	7
Integração de ferramentas	Integrar a ferramenta CVS com a ferramenta Y, para permitir a recuperação automática do registro de mudança que identifica a mudança	8
	Integrar as ferramentas disponíveis para a gestão dos projetos de software para facilitar o planejamento e acompanhamento.	1, 2, 3, 4, 5, 6, 9, 10
	Implementar a integração entre as ferramentas ClearCase e o RequisitePro com o objetivo de facilitar a interação entre atividades de Requisitos e a Gestão da Configuração do software	2
Necessidade de automatizar	Seria interessante adaptar a ferramenta CVS para verificar padrões...	6
	Poder-se-ia utilizar integração contínua para facilitar o controle dos itens.	6
	Otimizar a ferramenta X para que os consultores de GQS sejam notificados quando Y acontecer	8
	Automatizar mais o processo de software, evitando redundância e excesso de documentação	1, 2,3, 6
Disponibilidade e desempenho das ferramentas	Melhorar o desempenho das ferramentas de apoio ao processo, garantindo sua disponibilidade em tempo integral para todos os envolvidos	1, 2, 5, 9, 10
Necessidades locais x padronização/ Troca de conhecimento entre as unidades	Procurar eliminar a necessidade de procedimentos locais, desenvolvendo um trabalho de atualização do PSDS de forma a torná-lo mais adequado às exigências do CMM-SW e à realidade de suas unidades usuárias	2
	Incorporar ao PSDS o procedimento alternativo local para...	4
Tratamento de PMP	Perseguir o tratamento e retorno pontual das propostas de melhoria no processo (PMP), evitando o represamento de pedidos sucessivos, o que vem ocorrendo, e pode comprometer a credibilidade do processo.	2
	Maior agilidade no atendimento das propostas de melhoria no processo.	3
Evolução contínua	Implantar a macroatividade de Modelagem de Negócios	1, 2

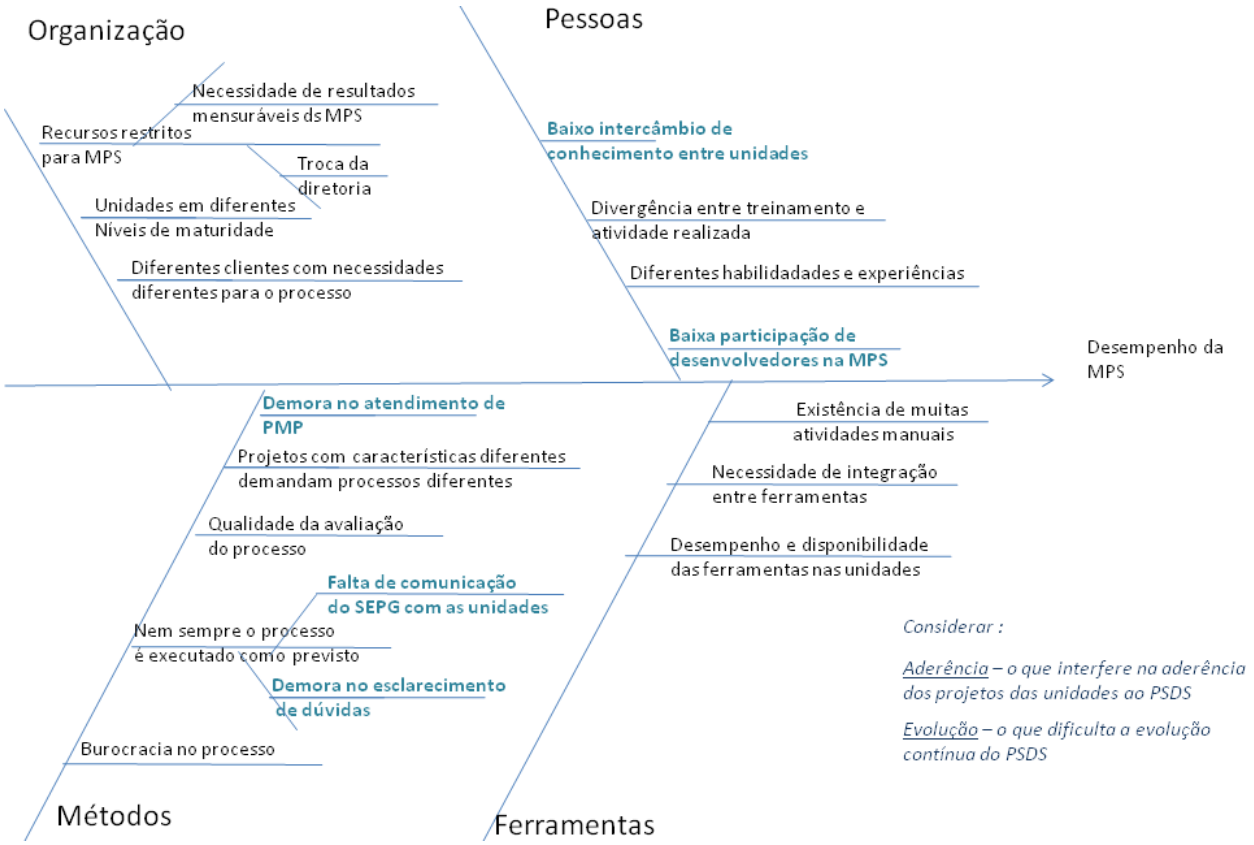


Figura 28: Diagrama de causa e efeito de fatores identificados para a MPS do SERPRO

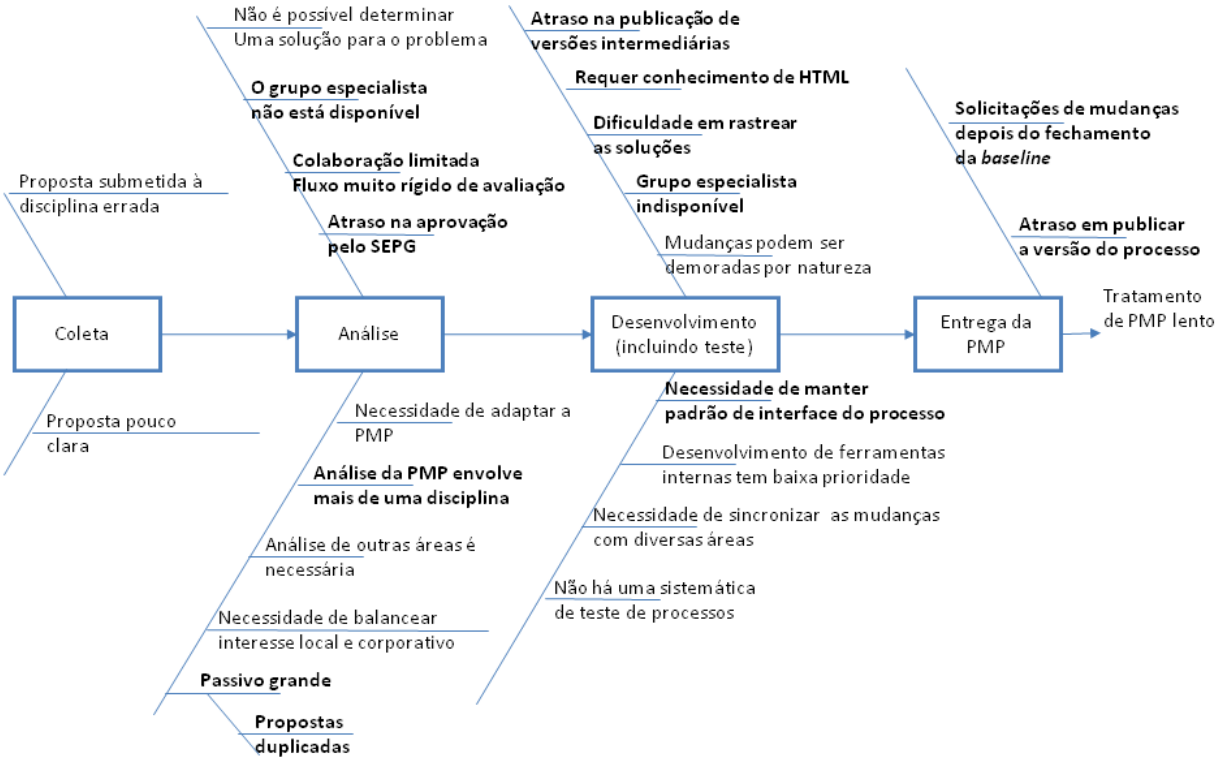


Figura 29: Análise de fatores para demora no tratamento de PMP

A tempestividade na resposta a uma dúvida é essencial para a aderência dos projetos ao processo. À medida que o número de usuários do processo cresceu, o atendimento às suas dúvidas foi ficando mais lento. Os fatores que poderiam influenciar a demora de esclarecimento de dúvidas dos usuários do processo de desenvolvimento de software ("pedidos de suporte") foram mapeados (Figura 30).

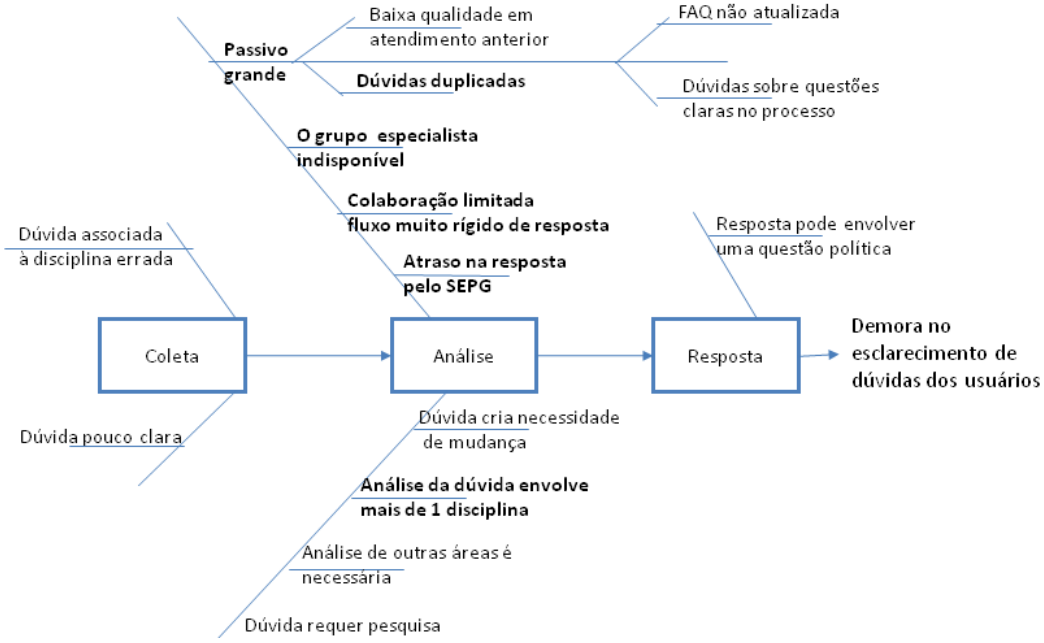


Figura 30: Análise de fatores para demora em esclarecer dúvidas dos usuários

Os fatores que podem implicar a baixa participação de desenvolvedores na MPS e nas questões relacionadas à comunicação entre unidades, e entre estas e o SEPG corporativo, já foram discutidos nas seções 3.3.4 e 3.3.5 (Figura 22 e Figura 24). Em ambos os casos, os fatores identificados no SERPRO foram apresentados em conjunto com os fatores encontrados na literatura relacionados à participação e motivação e à comunicação.

4.3 Experiência de uso de GC no desenvolvimento de software

Em 2000, o SERPRO criou um programa de Gestão de Conhecimento para preservar e evoluir o seu conhecimento organizacional. A estratégia de implementação da Gestão do Conhecimento na empresa envolveu a formação de comitês específicos nas unidades e a definição de políticas, metodologias e ferramentas para gestão de conhecimento. O programa, como foi concebido à época, não existe mais. Mas deixou alguns frutos relacionados à GC.

Nesta seção, são sintetizadas informações sobre a tecnologia associada à GC, os sistemas de informação utilizados e como o conhecimento é tratado no **desenvolvimento de software**. Para compreender esta seção é recomendada a leitura prévia das seções 4.1 e 4.2.

Mais informações sobre a estratégia de gestão do conhecimento e aprendizagem organizacional que foi adotada pelo SERPRO estão disponíveis em: Malheiros e Pinho (2003); Malheiros (2003); Santos (2001); SERPRO (2003); Carvalho et al. (2006).

A partir do portal corporativo interno do SERPRO, é possível acessar sistemas e serviços internos da empresa, como sistemas de recursos humanos ou financeiros. O portal disponibiliza acesso a: cursos de ensino à distância; documentos eletrônicos da empresa; listas de contatos da organização; ferramenta de busca; central de atendimento da empresa; normas; *sites* externos que possam ser de interesse dos funcionários; notícias internas e externas; agenda de eventos e *sites* internos temáticos por área de interesse, como desenvolvimento de software, planejamento empresarial ou relacionamento com clientes. O portal, assim como os portais internos temáticos, foi desenvolvido utilizando Zope/Plone[46], possuindo o mesmo padrão de interface.

A estrutura de gestão de conhecimento está disponível para os desenvolvedores, inclusive para elaboração/participação em curso de ensino à distância. As informações do PSDS estão disponíveis em um *site* na intranet do SERPRO que pode ser acessado a partir do portal.

No *site* do PSDS, estão mapeados os papéis e responsabilidades envolvidos no processo de desenvolvimento de software. Cada uma das atividades está detalhada em sub-atividades e estruturada em: definição, artefatos de entradas e saída e papéis envolvidos (Seção 4.2). A visualização hierarquizada das informações facilita o entendimento e torna a navegação amigável (Figura 26). A possibilidade de acessar informações correlatas por meio de hipertexto acelera o processo de aprendizagem, tornando-o mais interativo, personalizado e detalhado. Uma ferramenta de busca, que permite consultas não estruturadas, está disponível. Um glossário para consulta à terminologia do processo de desenvolvimento também pode ser acessado.

Existe um espaço (*Frequently Asked Questions* - FAQ) para que sejam registradas perguntas e respostas fornecidas por especialistas relacionadas a cada macroatividade do PSDS, responsáveis pela manutenção e evolução do mesmo. Esse espaço é atualizado pelos mantenedores do processo. Vale ressaltar que, com o passar dos anos, tem-se tornado difícil manter o FAQ de forma centralizada pelos mantenedores do processo e há desatualização.

[46] O Zope é um servidor de aplicações Web (http://www.zope.org/). O Plone é um sistema de gerenciamento de conteúdo (http://plone.org).

Visando a MPS, o PSDS oferece um espaço para que o indivíduo contribua com a otimização do próprio processo. Para o período de 2005 a 2008, propostas de melhoria de processo (PMP) podiam ser cadastradas por meio da solução GM-PSDS (Malheiros et al., 2006). A ferramenta implementa um fluxo de documentos (*workflow*) para tratamento das propostas e, em caso de dúvidas, é possível utilizar a GM-PSDS para registro de pedidos de apoio ao uso do processo. Para tratamento das FAQ, das PMPs e dos pedidos de apoio foram criados grupos de especialistas (MALHEIROS et al., 2006). Esses grupos resolvem também outros tipos de questões, como a definição de requisitos para ferramentas de desenvolvimento de uso corporativo (CARVALHO et al., 2006)

As informações específicas de **projetos de software** são disponibilizadas em repositórios de arquivos que são organizados com base na mesma taxonomia de organização das informações do PSDS. O desenvolvimento de software utiliza ferramentas de gestão de configuração e gerência de requisitos. As informações sobre os projetos ficam nos repositórios de gestão de configuração e algumas são consolidadas em lições aprendidas. Os artefatos e produtos ficam armazenados em um servidor central e são controlados por meio de ferramentas de gestão de configuração (Rational ClearCase e CVS).

Outras formas de disseminação do conhecimento utilizadas no SERPRO, relacionadas ao desenvolvimento de software, são: *workshops*, palestras, treinamentos, *mentoring* e educação à distância. O SERPRO conta com uma Universidade Corporativa que dá as diretrizes para o aprendizado organizacional. Com a migração do programa de melhoria para a Coordenação de Tecnologia e o fortalecimento da Universidade Corporativa, a gestão de treinamentos relacionados ao desenvolvimento de software passou para responsabilidade da universidade, incluindo os treinamentos do PSDS. Com isso foi possível centralizar o assunto treinamento em uma única unidade empresarial.

4.4 Considerações finais

Neste capítulo, foi apresentada a experiência em MPS, vivenciada pela autora, no SERPRO. Para compreender a maneira como a MPS acontecia, seus pontos fortes e limitações, dados foram coletados a partir de doze diferentes unidades de desenvolvimento da empresa. Dada a significativa influência de fatores “não-técnicos” na MPS, práticas de GC aplicadas ao desenvolvimento de software também foram identificadas.

Transformar uma PMP conceitual em uma melhoria aplicada na indústria não tem se mostrado uma tarefa fácil. Isso porque algumas influências ao sucesso, ou insucesso, de uma

solução não podem ser experimentadas em laboratório e, algumas vezes, são características de determinados contextos. Para identificar se as experiências do SERPRO poderiam ser aproveitadas em outros contextos, os fatores de influência para MPS identificados nesta experiência do SERPRO foram confrontados com os fatores de influência para MPS reportados na literatura (Capítulo 3), revelando que muitos dos desafios da MPS reportados eram semelhantes aos fatores identificados no SERPRO. Assim, espera-se que o conhecimento adquirido com o programa de MPS relatado nesse capítulo possa ser utilizado para o desenvolvimento de estratégias de MPS que possam ser reaproveitadas em outros contextos.

Algumas estratégias e mecanismos têm sido aplicados no SERPRO pela coordenação do PSDS, grupo do qual a autora deste trabalho faz parte. À medida que os estudos em MPS foram aprofundados novas estratégias e mecanismos foram definidos e testados em casos reais.

Tais estratégias e mecanismos, bem como estudos experimentais direcionados, são apresentados ao longo desta tese e, em particular, no Capítulo 6. As referências bibliográficas estudadas (Capítulo 2), o entendimento dos fatores que influenciam a MPS (Capítulo 3) e a experiência prática em MPS, apresentada neste capítulo, foram de grande valia para a proposição e aplicação da estratégia ColabSPI, apresentada no próximo capítulo.

5- ColabSPI - estratégia colaborativa e distribuída para MPS

Tendo em vista os fatores críticos para a MPS (Capítulo 3) e as observações práticas de um programa de MPS no SERPRO (Capítulo 4), neste trabalho, é proposta uma estratégia colaborativa e distribuída para MPS, denominada ColabSPI.

5.1 Considerações iniciais

Várias influências para o sucesso de programas de MPS estão relacionadas à coordenação, à comunicação e, principalmente, ao grau de participação e motivação em iniciativas de MPS. Cenários de desenvolvimento distribuído de software (equipes de desenvolvedores distribuídas) intensificam a necessidade de tratar essas influências. Nesses cenários, o processo é um fator crítico de sucesso (PRIKLADNICKI et al., 2004) e a variável dispersão torna processos para desenvolvimento distribuído mais complexos. Quanto mais complexo o processo, mais a sua melhoria será dificultada, já que a complexidade do processo dificulta sua evolução e a aderência a ele. Além disso, é preciso garantir a participação de desenvolvedores geograficamente dispersos na MPS.

À medida que o DDS se torna mais comum, a necessidade de uma estratégia distribuída e colaborativa para a MPS aumenta. Tal estratégia, bem como uma proposta de infraestrutura associada, tem por objetivo apoiar a MPS realizada por equipes heterogêneas e geograficamente dispersas. A ColabSPI oferece um referencial para conduzir programas de MPS, visando a: (i) identificar e gerir oportunidades de melhoria para o processo; (ii) evoluir, controlar e distribuir versões do processo; (iii) facilitar a institucionalização do processo em equipes distribuídas com diferentes níveis de maturidade; (iv) aumentar a participação dos desenvolvedores no programa de MPS; e (v) facilitar a comunicação entre os envolvidos. A adoção da ColabSPI também pode promover a criação de uma base de conhecimento sobre o processo de desenvolvimento de software e suas melhorias, eventualmente fornecendo lições aprendidas para a comunidade.

Influências para a ColabSPI foram: práticas adotadas para o DDS; práticas adotadas pelo desenvolvimento de SL; práticas de GC; soluções que apóiam o desenvolvimento de software

colaborativo – *Wiki*, ambientes de desenvolvimento colaborativos; e soluções de acompanhamento de erros (*bug tracking*).

O público inicialmente esperado para a ColabSPI são as organizações de grande porte que apliquem DDS e pretendam aplicar processos padrão para suas unidades de desenvolvimento. Há evidências de que seu uso pode ser útil em outros contextos, a exemplo de comunidades de software livre, que pretendam evoluir colaborativamente seus processos (Seção 6.3). Uma possibilidade a ser explorada também é a utilização da ColabSPI por um grupo de pequenas empresas, que compartilhem um processo de desenvolvimento de software, por exemplo, inspirado no MR.MPS (SOFTEX, 2009), e o evoluam conjuntamente.

Neste capítulo, apresentam-se: (i) uma visão geral da estratégia ColabSPI; (ii) seus quatro elementos, incluindo a infraestrutura para MPS associada; e (iii) considerações finais.

5.2 A construção da ColabSPI

Para formular a estratégia colaborativa e distribuída para a MPS, foram considerados fatores críticos de sucesso para a MPS, desdobrados em três itens: “Necessidades”, “Ações” e “Resultados”, formando o contexto da estratégia (Tabela 6). Investigando os problemas associados aos grupos de fatores, foram identificadas as principais necessidades relacionadas aos mesmos. Uma necessidade, se atendida, pode transformar uma fraqueza da MPS em um ponto forte, remover barreiras e mitigar riscos de insucesso. Em seguida, formas de tratar as necessidades foram identificadas e as ações necessárias foram relacionadas. Os resultados esperados da realização das ações também foram estabelecidos.

A fim de orquestrar uma solução que, em sendo adotada, pudesse contribuir para atender as necessidades identificadas e atingir os resultados desejados, a concepção da estratégia foi dividida em quatro elementos (Figura 31): (i) princípios, (ii) estrutura organizacional para MPS; (iii) principais etapas da MPS; e (iv) infraestrutura de MPS. Cada um desses elementos é detalhado nas seções 5.3 a 5.6, respectivamente.

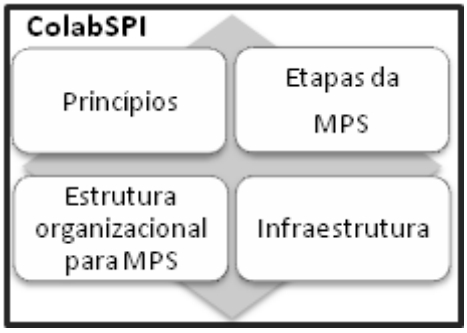


Figura 31: Componentes da estratégia ColabSPI

Tabela 6: Contexto da estratégia colaborativa e distribuída para MPS

	Melhoria Contínua	Aderência	Participação e motivação	Comunicação
Grupos-Fator	1. Natureza do processo de desenvolvimento (manutenibilidade e estabilidade do processo, erosão do processo) 2. Tratamento de PMPs 3. GC e estratégias de aprendizagem (flexibilidade na execução de tarefas) 4. Estratégia de implementação da MPS	5. Apoio e troca de conhecimentos (inclusive, troca de conhecimento entre as unidades) 6. Normas e procedimentos	7. Envolvimento dos empregados 8. Delegação de poderes 9. Apropriação do processo	10. Comunicação (com exceção do fator existência de experiência prévia ruim)
Necessidades	• Desenvolver e manter o processo como um ativo da organização (1) • Apoiar a manutenção do processo (1) • Gerenciar propostas de melhoria (2) • Criar equipes de ação e gerir alocação de recursos para MPS (3,4) • Promover alinhamento com objetivos estratégicos (4) • Gerenciar o programa da MPS (2,4) • Criar grupos para a MPS (1, 2, 3,4) • Introduzir mudanças iterativamente (2,4)	• Prover melhor entendimento do processo, organizando as informações de maneira estruturada (5,6) Fornecer apoio, esclarecendo dúvidas tempestivamente (5) • Promover troca de experiências entre áreas (5) • Identificar quem sabe o quê (5)	• Promover participação de desenvolvedores (7,9) • Apoiar melhoria colaborativa (7,9) • Dar mais autonomia aos desenvolvedores (8)	• Disseminar informações sobre o processo (10) • Promover *feedback* nos dois sentidos – grupo da MPS ↔ desenvolvedores (10) • Promover o entendimento dos objetivos de MPS (10) • Dar visibilidade do processo e da MPS (10)
Abordagem	• Utilizar solução para documentar o processo estruturadamente • Desenvolver e manter um site com a documentação do processo • Publicar o processo em *releases*, de acordo com um plano • Identificar PMPs e controlar seu fluxo até implantação • Criar um "projeto" para a MPS e planejá-la, prevendo entregas úteis e rápidas • Utilizar projetos-piloto para introduzir melhorias de maior impacto • Definir estrutura organizacional da MPS, incluindo os papéis e responsabilidades • Estabelecer grupos de trabalho (comunidades) para a MPS • Viabilizar trabalho "voluntário"	• Criar fórum para respostas a perguntas • Disponibilizar chat • Criar de Comunidades (ex.: grupos especialistas; SEPG locais) • Disponibilizar informação de quem sabe o que; quem é o grupo especialista, etc. • Utilizar uma solução para documentar o processo estruturadamente, incluindo as normas e procedimentos	• Identificar PMPs e controlar seu fluxo até implantação • Criar orientação sobre como o desenvolvedor pode contribuir • Criar listas de discussão sobre melhorias de processo • Disponibilizar versões de trabalho do processo, controlando configuração • Desenvolver e manter um site com a documentação do processo • Estabelecer grupos de trabalho (comunidades) para a MPS, com a participação dos desenvolvedores • Definir estrutura organizacional da MPS, focando a colaboração de desenvolvedores de diferentes unidades	• Planejar comunicação • "Portal" com acesso único, às informações relacionadas • Divulgar o plano da melhoria • Divulgar últimas notícias • Distribuir enquetes • Acompanhar dúvidas mais freqüentes em fóruns
Resultados	• Processo documentado em formato apropriado e versionado • Melhorias de processo gerenciadas • Projeto de melhoria planejado e acompanhado • *Releases* para as melhorias estabelecidas e processo sendo evoluído iterativamente	• Dúvidas esclarecidas mais rápida e adequadamente • Possibilidade de identificar quem sabe o quê em relação às macroatividades	• Listas de discussão ativas • Processo versionado • Orientações disponíveis • Desenvolvedores participando da MPS	• Desenvolvedores conhecem ações/novidades do processo • GPES conhece necessidade do desenvolvimento • Envolvidos sabem onde obter informações • Envolvidos conhecem o plano de melhoria

5.3 Princípios

Os dois principais focos da MPS são considerados: evolução e institucionalização. Ou seja, ColabSPI apoiará as atividades para evoluir o processo de desenvolvimento de software e as atividades para aumentar a aderência dos projetos a esse processo.

Os princípios que norteiam a ColabSPI são:

- A evolução dos processos acontecerá de forma iterativa. Esse princípio é compatível com a visão incremental do modelo CMMI (SEI, 2006) e do guia de implementação IDEAL (GREMBA e MYERS, 1997), duas referências para a MPS. Também é compatível com o princípio de entregas rápidas e úteis das metodologias ágeis de desenvolvimento de software. Essencialmente, esse princípio está relacionado aos fatores da categoria "MELHORIA CONTÍNUA", do grupo **estratégia de implementação da MPS** (Seção 3.3.2);
- Os usuários do processo de desenvolvimento de software participarão da sua evolução. Esse princípio foi estabelecido para atender aos fatores da categoria "PARTICIPAÇÃO E MOTIVAÇÃO" (Seção 3.3.4);
- A comunicação aberta e transparente será priorizada. As discussões, apresentações, artefatos de gestão, planejamento de novas versões e outros artefatos relacionados são públicos e facilmente acessíveis. Canais de comunicação são disponibilizados, garantindo que a comunicação relacionada à gestão da MPS seja pública e aberta. Esse princípio é importante em função dos fatores da categoria "COMUNICAÇÃO" (Seção 3.3.5) e pode aumentar a colaboração;
- O processo sendo melhorado buscará atender às necessidades locais (projetos de software) e às necessidades organizacionais. Esse princípio é uma premissa para o desenvolvimento de uma estratégia de MPS distribuída e está alinhado com o modelo MuNDDoS (PRIKLADNICKI, 2003) e o modelo de Maidantchik (1999); e

Além dos princípios gerais para a MPS, requisitos principais para a ColabSPI foram surgindo à medida que as necessidades foram desdobradas, e a abordagem minimamente delineada. Esses requisitos principais são listados na Tabela 7.

5.4 Estrutura organizacional para MPS

Por estrutura organizacional para MPS entende-se o conjunto de papéis, responsabilidades e organização das pessoas para executar as atividades relacionadas à MPS. O

CMMI (SEI, 2006) propõe como uma boa prática estabelecer equipes de ação de processos para implementar as ações da MPS. O CMMI aponta para a constituição de um grupo de engenheiros do processo de software (SEPG) como uma boa prática. Mas, apontar como esse grupo deve ser formado está fora do escopo do CMMI. A estrutura organizacional para MPS da ColabSPI propõe uma maneira de estabelecer tais equipes em organizações que trabalham com DDS.

Tabela 7: Requisitos principais para a ColabSPI

1	Disponibilidade de ferramentas de comunicação que permitam a cooperação, como, por exemplo, fóruns de discussão e listas de distribuição de mensagens de correio eletrônico. Adicionalmente, que permitam a comunicação de eventos, notícias e necessidades da MPS, a publicação de informações em quadros de notícias ou notificar comunidades com interesses específicos.
2	Acesso à informação acerca da MPS por meio de um ponto de acesso único.
3	Inclusão do processo de desenvolvimento em um mecanismo de controle de versão e permitir alterações no processo por mais de uma pessoa, em mais de um lugar.
4	Existência de uma estratégia colaborativa de MPS, com foco na delegação de autoridade e com orientações sobre como contribuir. Mecanismos colaborativos podem melhorar a disponibilidade de recursos.
5	Existência de um processo de tratamento de Propostas de Melhoria de Processo (PMPs) (fluxo de trabalho, papéis e funções) e a possibilidade de acompanhar a situação de cada proposta até a sua conclusão.
6	Apoio ao tratamento colaborativo de pedidos de suporte.
7	Acesso transparente às propostas de melhoria e pedidos de suporte registrados, permitindo a qualquer interessado opinar sobre as propostas e esclarecer dúvidas de outros usuários.
8	Existência de um espaço do usuário, com a disponibilidade de filtros de informação por usuário, possibilitando, por exemplo, a exibição de listas de PMPs submetidas pelo usuário ou por sua unidade.

Um fator que surge recorrentemente quando um programa de MPS é desenhado é garantir o comprometimento tanto de gerentes quanto dos desenvolvedores com a MPS. Se o contexto é de grandes organizações com unidades de desenvolvimento distribuídas, esse problema é ainda maior, por abranger não só a cadeia de comprometimento de cada unidade, como o comprometimento de cada unidade com uma visão estratégica organizacional, problema semelhante ao encontrado no DDS (PRIKLADNICKI, 2003). Por isso, é preciso que a estrutura organizacional para a MPS da ColabSPI apóie o desenvolvimento de ações locais, nas unidades, e, concomitantemente, ações organizacionais que direcionem a MPS como um todo. Para garantir o alinhamento das ações de MPS com os objetivos de negócio da organização e o fluxo de conhecimento tanto horizontal como vertical, a estrutura organizacional para MPS deve ser fortemente integrada à estrutura organizacional propriamente dita.

As observações no SERPRO (MALHEIROS, et al., 2006 e MALHEIROS et al., 2008a) apontam para a adoção de uma estrutura organizacional para MPS em camadas, fortemente

relacionada com os níveis gerencias da organização, favorecendo que as ações de melhoria sejam processadas nos diferentes níveis organizacionais. Para organizações distribuídas de software é importante que a estrutura organizacional seja formada por representantes locais e corporativos.

Na Figura 32, é apresentada a estrutura organizacional para MPS proposta para a ColabSPI. Nesta figura, é destacada a relação entre a estrutura organizacional de MPS, ao lado direito, e a estrutura organizacional administrativa, ao lado esquerdo.

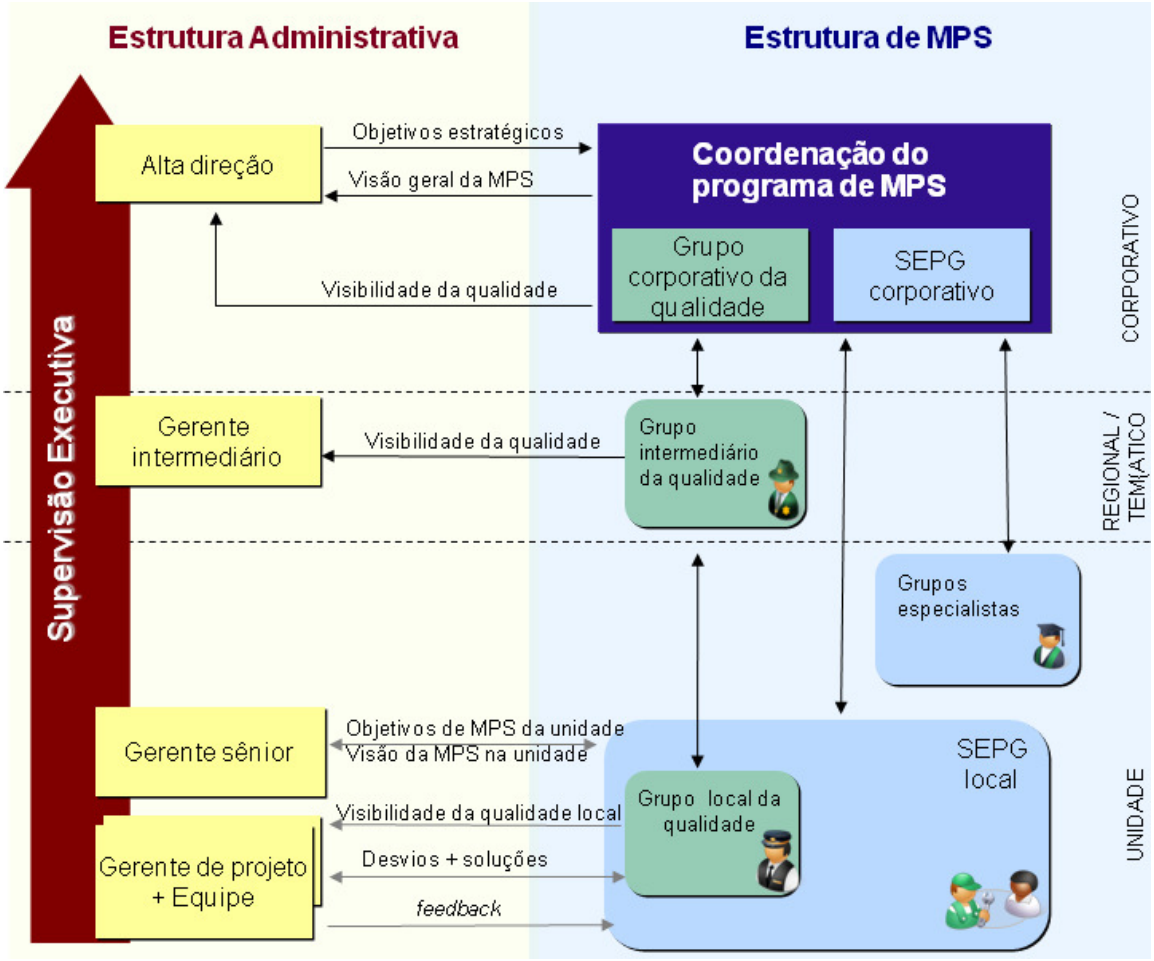


Figura 32: Estrutura organizacional para MPS da ColabSPI

A coordenação corporativa do programa de MPS (à direita na **primeira camada**, de cima para baixo, Figura 32) tem o objetivo de coordenar o programa de MPS e deve estar representada em níveis superiores da estrutura administrativa. Estar nesse nível traz credibilidade ao programa e proporciona visibilidade da organização como um todo. Para organizações cujo negócio principal é desenvolver software, este nível equivale à alta direção. Para as demais, corresponde à direção da unidade responsável por TI. Seu papel é garantir uma visão geral dos esforços investidos na MPS, agindo como facilitadora e direcionadora desses esforços, buscando colaboração e colaborando com as partes interessadas em MPS. Suas atividades principais são:

- Definir e manter vivo o programa de MPS, determinando diretrizes, estratégias e estrutura organizacional para MPS alinhadas com os objetivos estratégicos da organização;
- Acompanhar o uso do processo de desenvolvimento de software pelas unidades distribuídas desenvolvedoras de soluções de software e avaliá-las periodicamente;
- Gerenciar a aplicação de mudanças ao processo de desenvolvimento de software, acompanhando a implementação das melhorias no seu conteúdo e estrutura, e avaliando a adequação das mudanças, o grau de complexidade e benefícios para o processo, sempre com a diretriz de aumento da qualidade do software e da produtividade do desenvolvimento;
- Zelar pela simplicidade e praticidade na definição dos procedimentos e padrões, visando a melhorar a qualidade das soluções desenvolvidas e a produtividade das equipes;
- Buscar a integração do processo de desenvolvimento de software com os outros processos e projetos empresariais;
- Publicar o resultado das atividades da MPS para todas as unidades de software (MAIDANTCHIK, 1999);
- Fortalecer grupos de qualidade de software, como indutores do uso adequado do processo de desenvolvimento de software, enfocando a qualidade do produto final e não o uso do processo pelo processo; e
- Definir requisitos de ferramentas de apoio ao processo de desenvolvimento de software.

A coordenação do programa de MPS engloba dois grupos: (i) o **SEPG corporativo**, responsável pelas decisões estratégicas da melhoria; e (ii) o **grupo corporativo da qualidade**, responsável por orientar e orquestrar as atividades dos consultores de qualidade espalhados nas unidades, garantindo o alinhamento dos padrões de qualidade. A existência de um SEPG corporativo além de ter se mostrado uma boa prática no SERPRO (MALHEIROS et al., 2006), foi mencionada como uma boa iniciativa por Ogasawara et al. (2006). Vale ressaltar que a própria coordenação do programa de MPS pode trabalhar e tomar decisões de forma distribuída. A tomada de decisão distribuída relativa à MPS foi praticada no SERPRO. Em determinado momento o SEPG corporativo esteve distribuído entre Brasília, São Paulo, Rio de Janeiro e Ribeirão Preto.

A **segunda camada** está relacionada com o porte da organização e poderá existir, dependendo do tamanho da organização, da quantidade de unidades de desenvolvimento e de sua estrutura organizacional. Essa camada pode representar um agrupamento regional ou temático. Por exemplo, pode ser um agrupamento por países e consolidar os resultados por região das unidades de desenvolvimento. No caso do SERPRO, apresentado no Capítulo 4, essa camada intermediária foi utilizada para reunir unidades que desenvolviam software para um mesmo negócio (agrupamento temático). Na época, a empresa continha cerca de 40 unidades de desenvolvimento de software. Com a redução do número de unidades de desenvolvimento para 10 unidades (por meio da junção de algumas delas), a existência dessa camada está sendo reconsiderada. A existência dessa camada deve ser analisada caso a caso e ela pode ser muito importante. Mas, algumas vezes, pode ser melhor suprimi-la. Trabalhar com apenas duas camadas na estrutura organizacional da MPS pode implicar redução de níveis hierárquicos e aumento da comunicação informal, dois fatores que podem influenciar programas de MPS.

Os **grupos especialistas** integram a **terceira camada** e são indicados ou se voluntariam sem intermediários. A representação de diferentes unidades de desenvolvimento no grupo especialista favorece um processo que reflita a diversidade organizacional (tanto técnica quanto cultural) e aumenta a participação de diferentes desenvolvedores na definição e evolução do processo. Os grupos especialistas são os mantenedores dos ativos do processo. O foco de cada grupo é a evolução do processo no seu tema. Seu objetivo é apoiar o SEPG corporativo na análise de propostas de melhoria de processo, estimando ganhos e perdas de cada mudança e o tempo necessário para implementá-las. Se a segunda camada existir, os desenvolvedores das unidades locais que participam dos grupos especialistas são indicados pelos gerentes intermediários e não por cada unidade de desenvolvimento isoladamente.

Embora exista uma coordenação central da MPS, cada unidade de desenvolvimento (terceira camada**,** de cima para baixo na Figura 32) deve ser vista como uma organização autônoma que possui seus pontos fortes e suas deficiências, assim como suas expectativas em relação à melhoria do seu trabalho em particular.

Fazendo um paralelo com o modelo de DDS proposto por Prikladnicki (2003), a coordenação corporativa da MPS (primeira camada) está focada no planejamento estratégico da MPS e orquestra o trabalho das unidades distribuídas. Enquanto isso, cada unidade focará no desdobramento tático/operacional para sua área de atuação. São essas unidades que, de fato, desenvolvem o software e usam o processo de desenvolvimento de software.

Cada unidade distribuída de desenvolvimento tem um **SEPG local**, que promove o uso do processo internamente na unidade. O SEPG local está focado no uso local do processo, mas

também pode colaborar com a evolução global do processo, submetendo propostas de melhoria e pré-analisando as sugestões de desenvolvedores da sua unidade. Entre suas atribuições estão:

- Oferecer apoio técnico ao uso do processo para os projetos de software no seu âmbito de atuação;
- Acolher as sugestões da sua unidade organizacional;
- Garantir que o processo de desenvolvimento de software, adotado pelos projetos de software, tenham sido adaptados de forma apropriada, e podem contribuir para o desempenho do processo e para um produto com mais qualidade;
- Planejar e acompanhar o plano para implantação e melhoria do processo de software na sua unidade (MAIDANTCHIK, 1999);
- Incentivar a adoção de novas práticas; e
- Acompanhar freqüentemente o uso do processo em suas unidades.

A forma como se estruturam e o tamanho dos SEPG locais dependerão das características da unidade. A experiência no SERPRO mostrou que um arranjo adequado é trabalhar com ao menos o coordenador do SEPG local dedicado totalmente a essa atividade, em particular em equipes com mais de 50 desenvolvedores.

Os consultores de Garantia da Qualidade de Software (GQS) verificam a aderência dos projetos de software ao processo (centro da Figura 32). Cada unidade de desenvolvimento tem seu grupo de GQS coordenado por um GQS local que, periodicamente, dá visibilidade do uso do processo para o gerente daquela unidade e para o GQS Corporativo. O GQS local atua nos projetos em desenvolvimento apontando problemas de aderência ao processo que são considerados não-conformidades. Os outros níveis de GQS estão relacionados com a MPS distribuída e trabalham as questões que surgem quando é preciso orquestrar a MPS em unidades distribuídas e com diferentes níveis de maturidade. O GQS corporativo controla globalmente as atividades de GQS e supervisiona o trabalho do GQS das unidades, garantindo o alinhamento da qualidade entre as unidades e dando visibilidade da aderência ao processo aos altos níveis hierárquicos da organização. Se a camada intermediária (regional/temática) existir, é conveniente um GQS intermediário para consolidar os resultados para o gerente intermediário.

Os grupos de GQS dão o ritmo à institucionalização do processo nas unidades. É muito importante que seu foco esteja no uso do processo para aumentar a qualidade do software e o seu desempenho, e não no uso do processo pelo processo. Esse direcionamento poderá tratar alguns fatores para a MPS como redução da burocracia, proporcionar maior compreensão do processo de desenvolvimento de software, e, principalmente a reduzir a erosão do processo.

À medida que avaliam aderência ao processo, o GQS local identifica oportunidades de melhoria que devem ser reportadas, como por exemplo: (i) atividades no processo que geram desvios recorrentes, por não estarem claras ou bem definidas; e (ii) atividades burocráticas que não agregam valor à qualidade do produto final.

Uma importante questão relacionada à estrutura organizacional de MPS é em que proporção os envolvidos se dedicam à manutenção do programa (dedicação parcial ou total). Ter todos os envolvidos com MPS com 100% de dedicação pode aumentar o foco dessas pessoas nas atividades da MPS, já que, nesse caso, essa será sua atividade principal. No entanto, a dedicação exclusiva à definição de processos pode incorrer em processos inviáveis de serem executados e desconectados das necessidades reais do desenvolvimento, além de alienar os desenvolvedores, contribuindo para a execução do processo pelo processo.

A estrutura organizacional de MPS proposta na ColabSPI baseia-se em uma solução híbrida. Um pequeno grupo corporativo, SEPG corporativo, direciona as atividades de melhoria e é 100% dedicado às atividades do processo. O SEPG corporativo é apoiado por um grupo de desenvolvedores dedicados parcialmente ao desenvolvimento de software e, parcialmente, à MPS. Eles têm dedicação parcial às atividades da MPS e atuam prioritariamente como analistas de sistemas, projetistas, analistas de requisitos, gerentes de projeto, testadores, entre outros papéis no desenvolvimento de software. Tais desenvolvedores são a conexão com a linha de frente do desenvolvimento e podem trazer *feedback* sobre a adequação das novas versões do processo e sua efetividade. A experiência em trabalhar com equipes distribuídas apontou para um percentual de 20% de dedicação às atividades de MPS (em dias por semana) como um percentual adequado para balancear o foco na MPS e a conexão dos envolvidos com o dia-a-dia do desenvolvimento de software. Um efeito colateral positivo de trabalhar com equipes em dedicação parcial é que os desenvolvedores aplicam no desenvolvimento de um projeto o conhecimento obtido com o trabalho na MPS e tornam-se defensores da MPS. O fato dos desenvolvedores executarem o processo que eles ajudam a definir resulta em atividades mais focadas no desenvolvimento do software em si, diminuindo a demanda administrativa do processo e aumentando o intercâmbio de conhecimentos.

5.5 Etapas para MPS

Nesta seção, apresenta-se um ciclo de gestão para a MPS, que engloba as suas principais etapas (Figura 33).

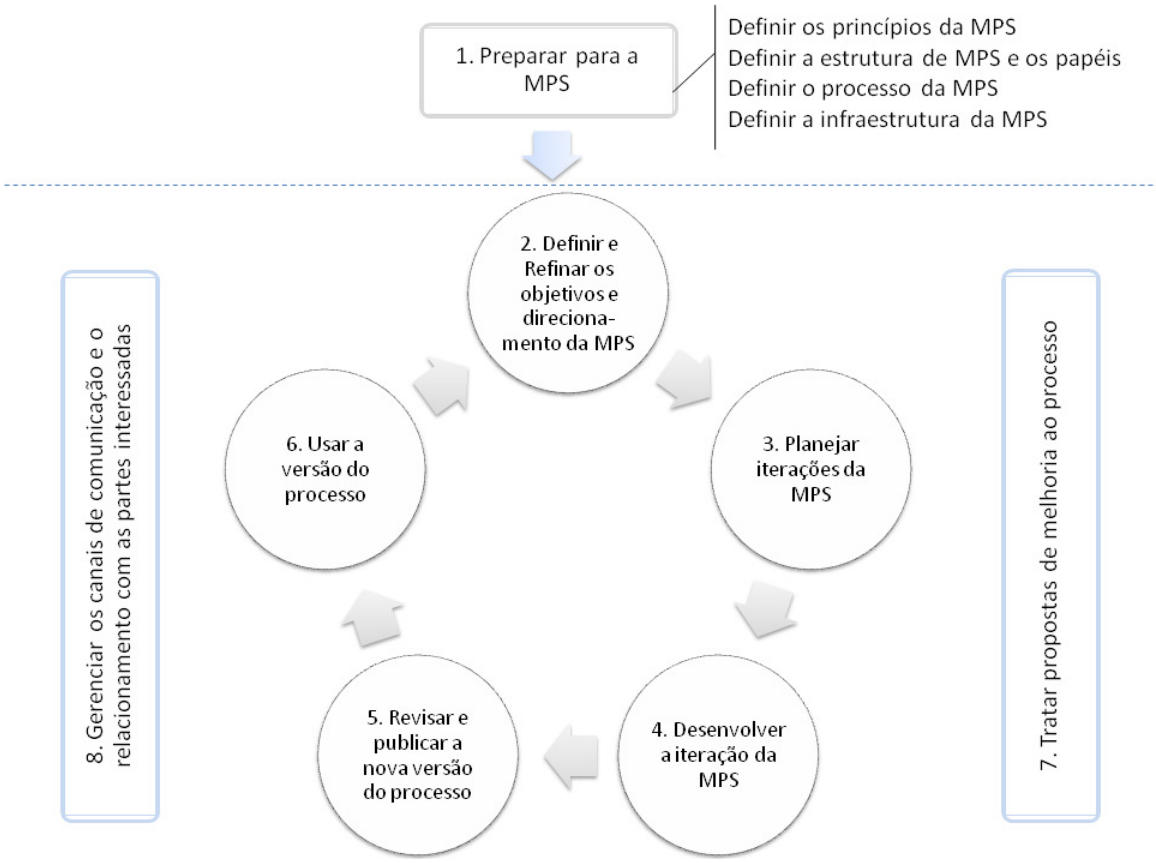


Figura 33: Etapas do ciclo da MPS

A evolução do processo deve ocorrer de forma iterativa (Princípio 1, Seção 5.2) em ciclos contínuos que englobam: definição, planejamento, desenvolvimento, liberação e uso do processo de desenvolvimento de software. Para apoiar esse ciclo da MPS, é preciso tratar as propostas de melhoria de processo, controlando as mudanças ao processo e gerenciando os canais de comunicação e o relacionamento com as pessoas envolvidas no ciclo da MPS.

Se o ciclo da MPS for comparado com o modelo QIP, pode-se afirmar que a maior parte das etapas do ciclo da MPS faz parte do ciclo de aprendizagem organizacional definido por Basili e Caldiera (1985), enquanto “usar a versão do processo” e algumas atividades de “tratar proposta de melhoria” fazem parte do ciclo de aprendizagem do projeto. No mesmo sentido, a maior parte das etapas está relacionada à evolução contínua do processo, enquanto “usar a versão do processo” está mais relacionada à aderência do processo.

As etapas para a evolução iterativa do processo são descritas nas seções 5.5.1 a 5.5.7. É possível estabelecer um paralelo entre essas etapas e o modelo IDEAL (GREMBA e MYERS, 1997). Além disso, já que a MPS acontece em um contexto distribuído, tais etapas estão alinhadas com as recomendações para a gerência da mudança do processo propostas por Maidantchik (1999).

5.5.1 Preparar para a MPS

A preparação para a MPS (Etapa 1, Figura 33) exige o conhecimento da organização onde a MPS está inserida. Nesta etapa, correspondente a etapa "*Initiating*" do IDEAL, é quando a base para empreender um esforço de MPS será montada, incluindo a definição dos objetivos estratégicos da organização para a MPS e a infraestrutura para alcançá-los.

Para executar o ciclo da MPS, é importante organizar as pessoas que se envolverão com a melhoria, definindo claramente papéis e responsabilidades. A Etapa "1" inclui a montagem da estrutura organizacional para MPS (Seção 5.4), a definição de seus princípios (Seção 5.3) e a preparação da infraestrutura de MPS (Seção 5.6). O responsável por essa etapa é o coordenador do programa de MPS.

Os resultados esperados desta etapa são: (i) a contextualização do programa de MPS no ambiente organizacional; (ii) a identificação da situação atual e dos grandes objetivos que irão nortear as decisões de MPS; (iii) a formalização da equipe de MPS; (iv) a definição dos princípios da MPS e do seu ciclo; e (v) a infraestrutura da MPS preparada.

5.5.2 Refinar os objetivos e definir o direcionamento da MPS

Os objetivos da MPS devem estar alinhados aos objetivos estratégicos da organização. Diferentes técnicas podem ser utilizadas para esse alinhamento, uma delas é o GQM$^+$ *Strategies* (BASILI et al., 2007). É preciso determinar o quê a organização espera obter como resultado da implantação de um programa de MPS. A organização deve realizar uma análise criteriosa da sua situação atual em relação ao desenvolvimento e manutenção de software, identificando quais os pontos deseja aprimorar. Além disso, é preciso determinar os objetivos da MPS claramente e divulgá-los para todos os envolvidos para o adequado direcionamento do programa de MPS, uma vez que, antes de se iniciar um movimento de mudança, é preciso, primeiramente, conhecer qual o destino no qual se deseja chegar.

Essa etapa é compatível com as etapas de diagnóstico e de planejamento do IDEAL. Como a identificação de PMPs é uma atividade contínua que acontece durante todo o ciclo, o diagnóstico realizado nessa etapa é simplificado e a maior parte das atividades está relacionada com a etapa planejamento do IDEAL.

Estudos apontam para o comprometimento da alta gerência como fator crítico de sucesso para a MPS (ABRAHAMSSON, 2001; SANTOS et al., 2007). Manter esse comprometimento e repassá-lo para a equipe é fundamental (NASCIMENTO e MALHEIROS, 2004). A motivação de baixo para cima também é possível e deve ser perseguida, mas tende a ser mais demorada e

desgastante. Sem o necessário comprometimento da alta gerência, em muitos casos, o sucesso de um programa de MPS pode ficar comprometido (BADDOO et al., 2007).

Uma vez delineados os objetivos da MPS, esta poderá ser conduzida como um projeto (APPLETON, 1997) e, nesse caso, o próximo passo é estabelecer o escopo e os produtos que serão gerados. O escopo pode ser documentado de várias formas. Uma opção para estruturar o escopo é utilizar uma Estrutura Analítica do Projeto – EAP - (em inglês, *Working Breakdown Structure - WBS*). A EAP é uma técnica de decomposição do trabalho do projeto em partes gerenciáveis. Os produtos de um projeto podem ser organizados (ou decompostos) de diferentes maneiras.

No contexto da MPS, pode ser oportuno decompor a partir de assuntos típicos da MPS, separando ao menos a evolução do processo do seu uso. Observações na prática indicaram que esses assuntos podem ser: (i) Evolução; (ii) Aderência (relacionada ao uso adequado); (iii) Gestão da MPS; (iv) Ferramentas; (v) Comunicação e Comunidade. Um exemplo de EAP elaborada para gerir um programa de MPS é apresentado no Apêndice A. A depender da necessidade organizacional, o assunto Integração pode ser acrescentado para englobar melhorias de integração com outros processos corporativos. Sugere-se orientar a EAP por entregas para facilitar o estabelecimento do plano corporativo do projeto. Caso a equipe do projeto tenha experiência em metodologias ágeis, pode ter mais facilidade de tratar o escopo da MPS como um *product backlog* do SCRUM (SCHWABER, 2006), em complemento à EAP.

A partir da EAP, pode-se desenvolver um plano de alto nível do projeto, um *roadmap* da MPS, que deve ser estabelecido e mantido pela coordenação do programa de MPS. O *roadmap* dá a visão geral do que se espera da MPS, enquadrando o escopo em entregas organizadas em uma linha do tempo com marcos gerais. O ideal, para manter o programa de MPS vivo na organização, é possuir entregas curtas e úteis, ainda que essas entregas não sejam necessariamente versões do processo. Uma entrega pode ser, por exemplo, um treinamento executado. O desenvolvimento e entrega dos produtos previstos na EAP e organizados no *roadmap* serão detalhados no planejamento de cada iteração.

Além de definir um plano corporativo para a MPS, que deve ser divulgado para todas as unidades distribuídas, será necessário que cada unidade defina o seu plano local para a MPS.

5.5.3 Planejar iterações da MPS

Essa etapa está inserida na etapa planejamento do modelo IDEAL. Para maior agilidade na MPS, o planejamento de uma iteração será feito iterativamente e só deve ser detalhado no início de cada iteração de melhoria.

A entrega do processo, tal qual a entrega de um software, deverá acontecer em *releases*. A política de publicação de versões do processo de desenvolvimento de software deve considerar a disponibilização de *releases* planejadas e de *releases* eventuais. Um exemplo de política de publicação está disponível no Apêndice A.

Para desenvolver essas *releases*, iterações de evolução do processo devem ser planejadas. É importante que os interessados na MPS participem desse planejamento. Alcançar um nível de maturidade é uma tarefa de todos, sendo assim, enquanto houver pessoas desmotivadas existirão barreiras e dificuldades em alcançar plenamente a institucionalização do processo na organização (NASCIMENTO e MALHEIROS, 2004). Uma boa estratégia para obter comprometimento é criar as condições necessárias para o envolvimento das pessoas nas tarefas de planejamento e implementação da MPS. Questionários e entrevistas de feedback e avaliação do processo podem ser insumos importantes para o planejamento. Por meio da participação, as pessoas tendem a se sentir mais engajadas e responsáveis pelo resultado final. Segundo Rainer e Hall (2002), *process ownership* é um dos fatores críticos de sucesso da MPS.

É nessa etapa que acontecerá o detalhamento do *roadmap*, estabelecido na Etapa 2 do ciclo. O escopo da iteração e quais os produtos e atividades serão realizadas devem ser divulgados para que os desenvolvedores possam contribuir com as melhorias.

5.5.4 Desenvolver iteração da MPS

Nessa etapa, correspondente à etapa de ação do modelo IDEAL, as atividades planejadas serão realizadas, desenvolvendo o escopo previsto para a iteração. A solução para as PMPs previstas para a iteração é concebida e implementada. Se a mudança desenvolvida alterar significativamente o processo de desenvolvimento de software é importante experimentá-la em projetos-piloto, antes de publicar em uma nova versão do processo. O uso de projetos-piloto para a MPS é sugerido pelo modelo IDEAL e discutido na Seção 6.2.4

5.5.5 Revisar e publicar nova versão do processo

Uma vez desenvolvidas as mudanças para o processo, é preciso revisá-las. Ou seja, é preciso testar a nova versão do processo. Se as mudanças estão implementadas, conforme previsto, e a nova versão do processo está consistente, o processo deve ser publicado, seguindo o planejamento realizado na Etapa 2 e a política de publicação do processo.

A publicação de uma nova versão para o processo marca a transição do ciclo de aprendizado organizacional da MPS para o ciclo de aprendizado local, que será executado nas unidades de desenvolvimento.

Cada publicação deve ser divulgada, de acordo com suas características, seguindo o plano de comunicação da MPS. Um exemplo de uma entrada para o plano de comunicação está na Tabela 8, com destaque na primeira linha para as informações que o plano deve conter.

Tabela 8: Informações em um plano de comunicação

Informações a Divulgar	**Forma de Divulgação**	**Periodicidade**	**Responsável**	**Público Alvo**
Divulgação de novos *releases* programados do processo	Informe enviado para a lista de e-mail, Portal do PSDS e espaço para últimas notícias	Conforme Planejamento de release	Maria	Desenvolvimento, Diretoria, Grupos de MPS

Além da nova versão, deve ser publicada a lista de PMPs que estão sendo atendidas com a nova versão, os impactos para os projetos de não migrar para a nova versão e, se for o caso, o material de treinamento associado às novas atividades.

É importante que as versões anteriores do processo continuem disponíveis para utilização pelos projetos que estão em andamento e para manutenção da base histórica do processo.

5.5.6 Usar a nova versão do processo

Esta é a única etapa do ciclo que é executada exclusivamente nas unidades de desenvolvimento e está inserida na etapa de ação do modelo IDEAL. O SEPG corporativo apenas acompanha o uso do processo.

É o uso do processo nos projetos de desenvolvimento de software que alimentará a MPS. Enquanto usa o processo, os desenvolvedores identificam e propõem PMPs que serão tratadas pela estrutura organizacional de MPS, contribuindo para o ciclo de aprendizado organizacional e realimentando o ciclo de evolução do processo.

5.6 A infraestrutura de MPS

O principal objetivo da infraestrutura de MPS é favorecer o surgimento, crescimento e continuidade de um ambiente colaborativo e distribuído para a MPS, facilitando a troca de experiências e o compartilhamento de conhecimento entre os envolvidos na MPS. Tal infraestrutura de MPS foi concebida para apoiar as boas práticas e as principais atividades da MPS, estando alinhada com:

- O CMMI (em especial as áreas de processo OPD - *Organizational Process Definition* - e OPF - *Organizational Process Focus*);

- As características de iteração contínua e uso de lições aprendidas propostas pelos principais guias de implementação de MPS (ex.: IDEAL (GREMBA e MYERS, 1997) e Fábrica de Experiências (BASILI et al. 1994b));
- As práticas de GC; e com
- Os procedimentos, práticas e ferramentas já aplicados ao desenvolvimento de software, em particular ao DDS e ao desenvolvimento de SL.

O elemento infraestrutura da ColabSPI pode ser materializado de diversas formas, tanto por meio da adaptação e/ou integração de soluções voltadas para o desenvolvimento de software, quanto por meio da concepção de soluções específicas.

Dyba (2001) havia ressaltado, e foi ratificado pela pesquisa na literatura relacionada à MPS, que muitos dos fatores que influenciam a MPS estão relacionados à gestão de conhecimento. Por esse motivo, a concepção da ColabSPI foi inspirada no modelo de GC aplicado ao desenvolvimento de software (LINDVALL et al.,2002) e com adaptações para apoiar a MPS. O modelo de MPS, sob a perspectiva de GC, está representado na Figura 34.

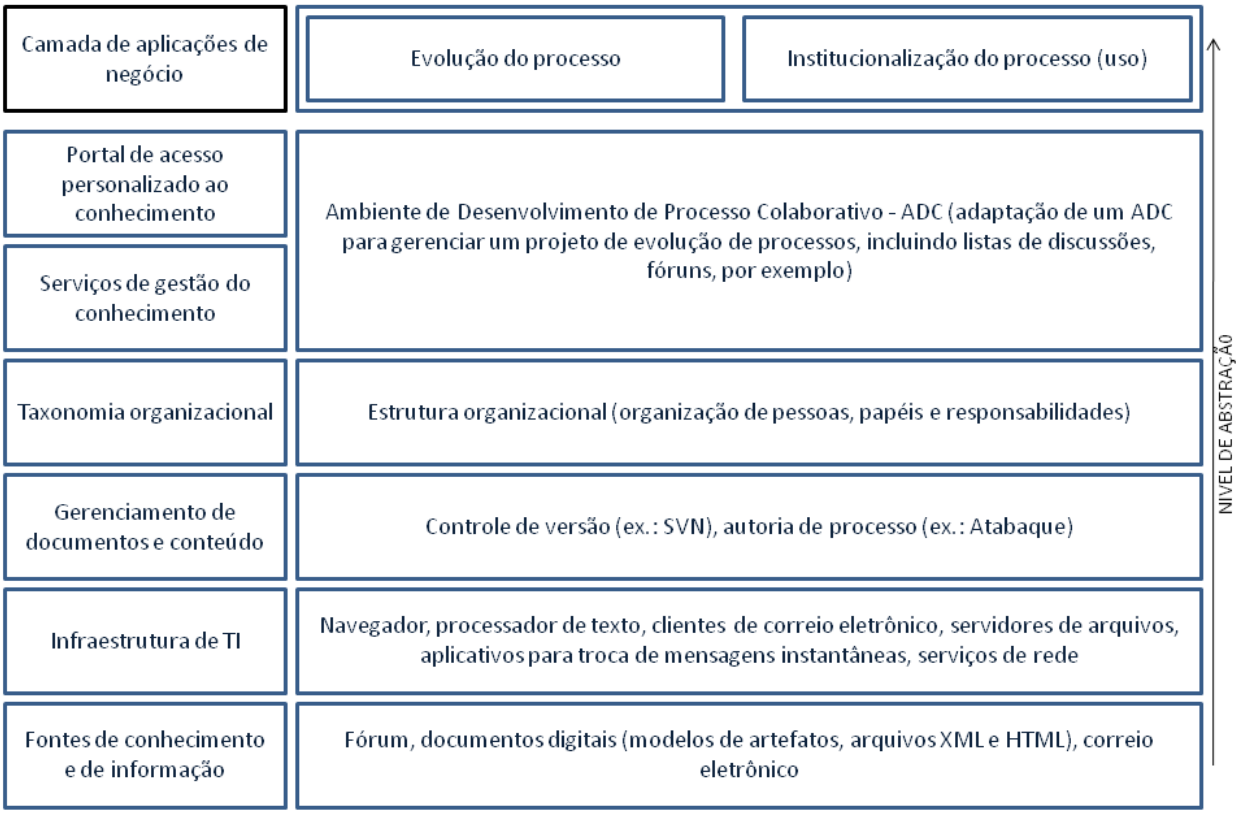


Figura 34: GC na ColabSPI. Adaptado de Lindvall et al. (2002) para a MPS

A primeira camada de baixo para cima (Figura 34) representa repositórios de documentos como correio eletrônico, registros em bancos de dados e documentos digitais. São exemplos de fontes de conhecimento e informação planos de gestão da MPS e arquivos HTML e XML que

explicitam o próprio processo de desenvolvimento de software. O conhecimento da MPS explicitado na camada “Fontes de Conhecimento e de Informação” será capturado por meio da camada “Infraestrutura de TI”. Nesse sentido, um navegador para páginas HTML pode ser utilizado para consultar o processo ou um processador de texto pode ser utilizado para editar o plano de gestão da MPS. Tal conhecimento poderá ser organizado, versionado e recuperado por meio de gerenciadores de documentos e de conteúdo, que oferecerão acesso organizado ao processo de desenvolvimento de software. A camada de taxonomia organizacional, no caso da MPS, corresponde à definição de uma estrutura organizacional que indica como as pessoas se organizam, quais são os papéis e as responsabilidades específicos para a MPS (Seção 5.3).

A próxima camada, de baixo para cima, aglutina serviços de gestão do conhecimento que disponibilizam apoio à colaboração, ou registro da memória organizacional (redes de especialistas, bases de soluções e de lições aprendidas). Em relação a esse nível de abstração da gestão do conhecimento, a ColabSPI apóia atividades de colaboração e comunicação integradas por meio de um ADC. Todos esses serviços são oferecidos por meio de um portal de acesso único ao conhecimento, que em última instância viabiliza a execução dos dois principais negócios (foco) da MPS: “evolução do processo” e seu “uso” (ou institucionalização).

Para compor a ColabSPI, procedimentos, práticas e ferramentas já aplicados ao desenvolvimento de software foram considerados, sempre sob a perspectiva de suas possíveis contribuições para a melhoria contínua de processos. Dentre esses, destacam-se: mecanismos para controle de versão; ferramentas para tratamento de erros em software, mecanismos para comunicação e coordenação de atividades. A aplicação de tais recursos foi considerada em três grupos funcionais (Figura 35):

- Documentação e evolução do Processo, ou seja, definição e melhoria do processo em si (Seção 5.6.2);
- Tratamento de Propostas de Melhoria (Seção 5.6.3);
- Comunicação e Colaboração (Seção 5.6.4).

Boa parte das atividades representadas na arquitetura da ColabSPI já é apoiada, de alguma forma, por serviços ou aplicações existentes no domínio do desenvolvimento de software. Sempre que possível, os serviços ou aplicações existentes foram reaproveitados para o domínio da MPS, com maiores ou menores adaptações. Para contemplar algumas atividades e lacunas, optou-se por desenvolver soluções específicas para a MPS. A arquitetura de software proposta para a ColabSPI é apresentada na Seção 5.6.1.

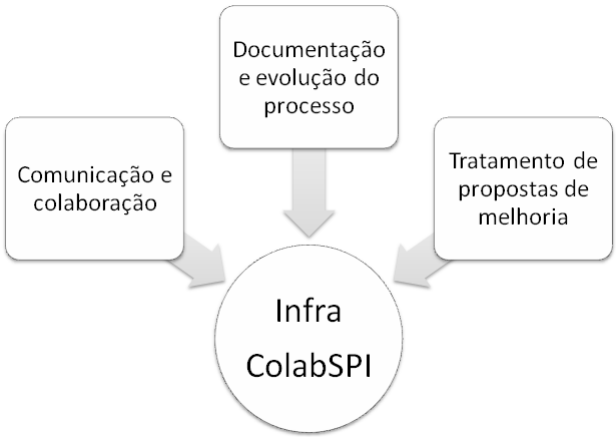


Figura 35: Organização das estratégias da ColabSPI por grupo funcional

5.6.1 Arquitetura da infraestrutura da ColabSPI

Para facilitar a organização dos mecanismos, aplicações e repositórios de informação de apoio à MPS em uma infraestrutura, foi desenhada uma arquitetura de software que visa auxiliar na compreensão de como esses mecanismos podem se integrar, de quais são as atividades suportadas pela ColabSPI e de como a infraestrutura pode evoluir de maneira incremental.

A arquitetura da infraestrutura da ColabSPI (Figura 36) foi inspirada na arquitetura de referência proposta por Nakagawa (2006) - RefASSET, que visa apoiar o projeto arquitetural de ambientes e ferramentas de Engenharia de Software. A escolha da RefASSET como referência foi motivada por esta possibilitar o projeto de ambientes integrados de forma incremental e por ser compatível com as características web projetadas para a infraestrutura da ColabSPI. A RefASSET é orientada a aspectos e propõe a separação de interesses em *frameworks* transversais. A arquitetura da infraestrutura da ColabSPI seria de mais valia se a infraestrutura fosse toda desenvolvida desde o início. Nestes casos, as propostas de orientação a aspectos e de usar *frameworks* transversais poderiam ser implementadas como sugerido. De todo modo, a visão organizada, com separação de interesses, da arquitetura RefASSET foi importante para modelar a arquitetura da ColabSPI e projetar uma visão de como mecanismos, aplicações e repositórios existentes podem ser organizados para a MPS.

A infraestrutura da ColabSPI deve ser disponibilizada na plataforma web e apresentar características de aplicações hipermídia, servindo simultaneamente a diversos usuários distribuídos. O uso de uma solução web aument a flexibilidade da solução e facilitar a integração de novas funcionalidades.

O lado cliente (Figura 36) corresponde à interface gráfica que possibilita ao usuário interagir com as atividades de MPS por meio de um navegador *web* e de uma ferramenta de

autoria do processo (Seção 5.6.2). No lado servidor, estão as camadas de apresentação, aplicação e persistência.

A camada de apresentação é responsável pela interface do usuário e gerencia esta interface, contendo serviços como controle de sessões e gerenciamento do que será mostrado para o usuário. No caso da ColabSPI, boa parte desses serviços está encapsulada pelo Ambiente de Desenvolvimento Colaborativo (ADC), que viabiliza o acesso às atividades da camada de aplicação. Afora as atividades disponibilizadas pelo ADC (Seção 5.6.4), o usuário tem acesso direto à versão corrente do processo propriamente dito, publicada em formato HTML, tanto por meio do navegador como por meio da ferramenta de autoria.

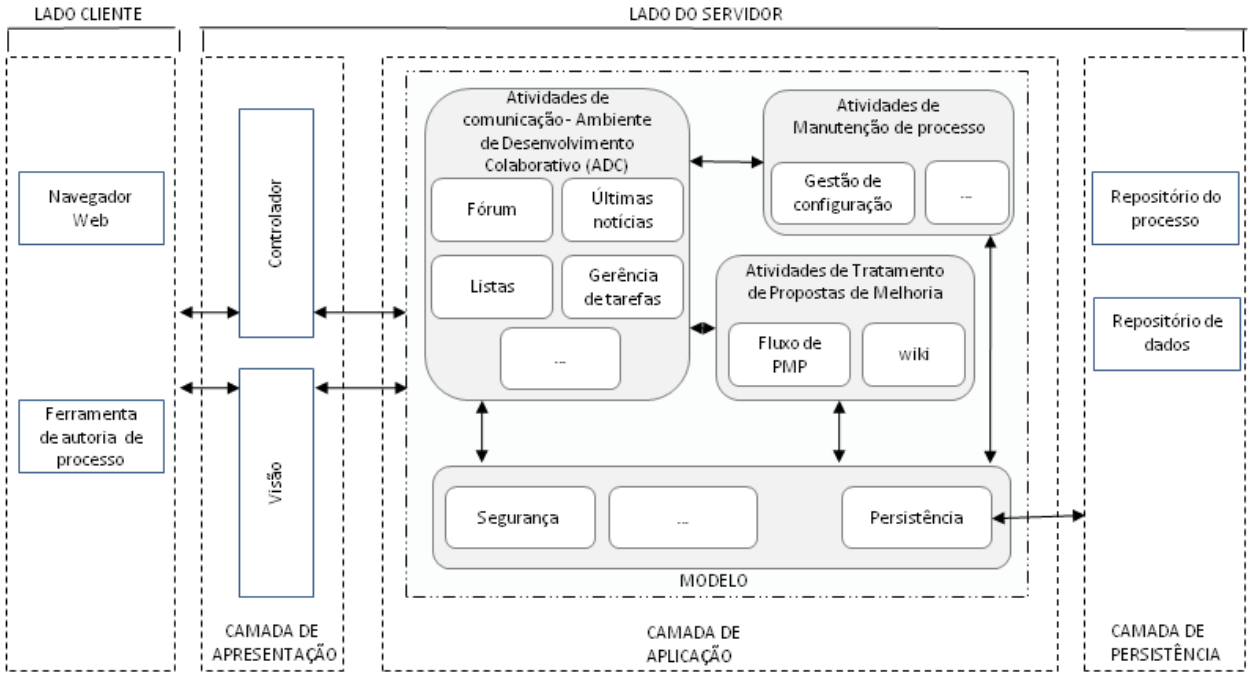


Figura 36: Arquitetura da infraestrutura da ColabSPI

A camada de aplicação contém as principais funcionalidades, ou seja, a lógica do negócio. Os grupos de atividades relacionadas ao "negócio" MPS foram organizados de acordo com três grupos funcionais da camada de aplicação (Figura 36) propostos para a ColabSPI. Cada um destes grupos será descrito nas seções 5.6.2 a 5.6.4, respectivamente: atividades de documentação e evolução do processo, atividades de tratamento de PMPs e atividades de comunicação. Neste trabalho, as atividades de manutenção de processo consideraram a documentação de processos descritivos. Mas, a aplicação da infraestrutura para processos automatizados também pode ser considerada, em particular as atividades de comunicação.

Dados consumidos e produzidos pela camada de aplicação devem ser armazenados em repositórios de dados e de processo da camada de persistência para posterior utilização. O papel

da camada de persistência é gerenciar os elementos persistentes da aplicação. Além do processo em si, que deve ser armazenado, tanto os dados de comunicação, quanto os dados do tratamento de PMPs, armazenados em bancos de dados, devem ser persistidos.

Funcionalidades da ColabSPI necessitam de mecanismos de controle de acesso para seu uso (vide "Segurança" na Figura 36). Diferentes papéis possuem diferentes níveis de acesso na ColabSPI, mas qualquer usuário pode consultar o processo e algumas informações públicas da MPS, sem necessariamente, se identificar. Aspectos de segurança ou persistência já são implementados pelas soluções utilizadas para montar a infraestrutura da ColabSPI, mas caso não estivessem disponíveis poderiam ser implementados como aspectos, como proposto por Nakagawa (2006).

Vale ressaltar que não é objetivo desta tese discutir em profundidade arquiteturas de referência de software. Um consolidado sobre o assunto e exemplos de utilização pode ser encontrado em Nakagawa (2006).

5.6.2 Documentação do processo

Além de definir um processo, para que ele se mantenha adequado, é preciso melhorá-lo. No entanto, a manutenção eficiente do processo e da sua documentação, garantindo a integridade do mesmo, pode ser uma atividade onerosa, complexa e propensa a erros. Tal qual acontece nas manutenções evolutivas e adaptativas de software, é possível identificar e tratar fraquezas de uma versão de um processo de desenvolvimento de software, convertendo-as em melhorias para a próxima versão (Figura 37). Assim como ocorre no desenvolvimento de software, soluções automatizadas que apóiem a documentação e a evolução de processos podem ser muito úteis.

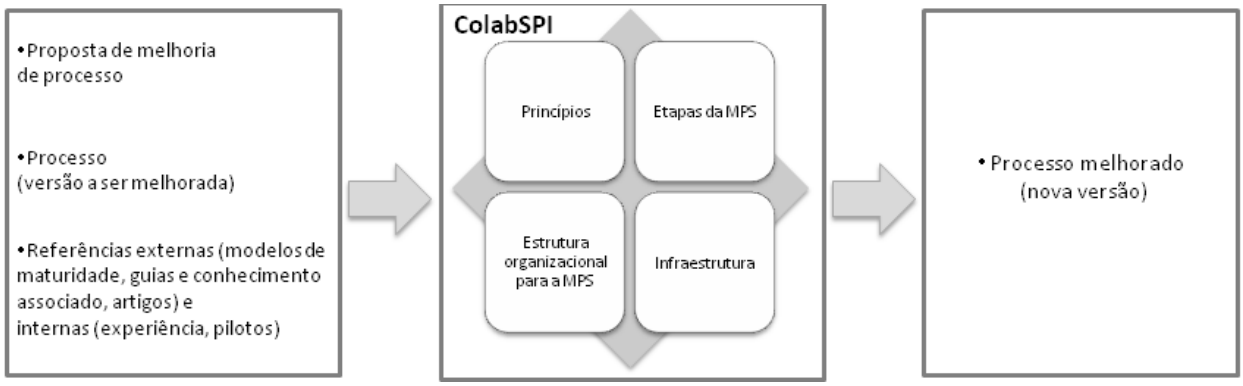


Figura 37: Contexto da documentação e evolução de processo documentado

As premissas para a documentação do processo, na ColabSPI, foram: (i) cada vez mais, organizações utilizam processos eletrônicos (MOE e DYBA, 2006); (ii) processo documentado deve estar acessível para todos os interessados; (iii) a documentação do processo é controlada em

versões e publicada em *releases*, à medida que o processo evolui. O SEPG corporativo decide o escopo de cada *release*, coordenando o trabalho dos contribuidores do processo; (iv) o conteúdo do processo é documentado de forma estruturada, facilitando o reúso de componentes do processo; e (v) características de usabilidade são observadas.

Uma outra questão considerada para a documentação do processo na ColabSPI foi o foco na descrição do processo em si.

As pesquisas na área nos últimos anos têm explorado duas principais vertentes (ARBAOUI et al., 2002 apud BERTOLLO, SEGRINI e FALBO, 2006): (i) abordagens para modelagem, análise e melhoria de processo de software; ou (ii) tecnologia de apoio ao processo de software (Seção 2.2). Em ambas as vertentes, uma descrição efetiva dos ativos do processo pode ser decisiva para a compreensão tanto do objetivo de cada componente isoladamente (atividade, papel, etc.) quanto do fluxo a ser adotado (integração dos componentes) (MALHEIROS et al., 2008b). Nesse sentido, documentar e evoluir o processo em um formato mais descritivo é útil inclusive para os ambientes de engenharia de software baseados em processos (Seção 2.2.), que normalmente discutem a evolução de processos no âmbito de melhorias para a sua automatização, mas não destacam as questões relacionadas à evolução da descrição associada ao fluxo do processo automatizado. Santos et al. (2007) mencionam que a MPS pode ser facilitada por uma infraestrutura de processos de software e se referenciam a um ADS completo, a Estação TABA, como uma opção para ser essa infraestrutura, apresentando resultados positivos da utilização da estação. Aparentemente, a MPS aqui discutida com o foco na descrição dos elementos do processo poderia ser aproveitada pela Estação TABA, por exemplo.

A definição e manutenção de processos são tarefas inerentemente baseadas em conhecimento e suscetíveis a vários tipos de problemas, agrupados neste trabalho em duas categorias - problemas semânticos e problemas de exibição (MALHEIROS et al., 2008b).

Os **problemas semânticos** são aqueles relacionados com o conteúdo, o significado de cada ativo do processo. A utilização da expressão “problema de conteúdo” será evitada para que não se confunda com o significado atribuído à palavra “conteúdo” na expressão “geração de conteúdo”, comum em soluções de documentação em hipermídia. São exemplos de problemas semânticos: definição equivocada de termos; fórmulas de cálculo erradas; definição de procedimentos burocráticos e pouco eficientes; e inconsistência interna.

Os **problemas de exibição** são aqueles relacionados com a maneira como os ativos documentados do processo são disponibilizados. Estão nessa categoria erros como: tamanho e

tipo de fonte; *links* quebrados e inconsistentes; alinhamento; cores; e consistência entre a forma de exibição e o conteúdo semântico que está sendo exibido.

Os dois tipos de problemas precisam ser tratados e, paulatinamente, eliminados do processo. Ambos demandam tempo e recurso para serem tratados e podem influenciar negativamente na qualidade do processo e, conseqüentemente, do produto final. Apesar de serem de mais simples identificação e correção, os problemas de exibição podem comprometer a obtenção de conhecimento a partir do processo de desenvolvimento de software documentado e promover resistência à sua utilização, além de gerar descrédito no conteúdo disponibilizado. Problemas de exibição podem impactar a usabilidade do processo.

O interesse em usabilidade cresceu após o gradativo reconhecimento do quanto às *interfaces* são mal projetadas e do quanto uma *interface* elegante pode trazer de benefício (SHNEIDERMAN e PLAISANT, 2004). A norma NBR/ISO 9241-11 (ABNT, 2002) define usabilidade como a medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico. Questões de usabilidade do processo documentado ainda não são extensivamente discutidas na literatura. É provável, no entanto, que a usabilidade de um processo possa influenciar na absorção do conhecimento contido no processo.

Além da documentação do processo, sua evolução controlada é essencial. Nesse sentido, é interessante que o processo esteja sob controle de versão. Um sistema de controle de versão é a combinação de tecnologias e práticas para rastreamento e controle de mudanças em um arquivo, que, no contexto deste trabalho, é um arquivo de processo. Na engenharia de software, e, em particular na ISO/IEC 12207 (ISO/IEC, 2008), o controle de versões é uma atividade associada à **gerência de configuração**, um conjunto de atividades para administrar modificações nos itens de informação ao longo do ciclo de vida do software, evitando, assim, que haja inconsistência entre os diversos itens de configuração de software. A gerência de configuração deve apoiar: (i) o controle de versões e as publicações em versões; (ii) a estabilidade dos arquivos (ativos do processo); (iii) e a comunicação entre os responsáveis pela evolução do processo. No módulo "Documentação e evolução do processo", o objeto principal da gestão de configuração é o processo em si. Elementos do processo, tais como papéis, atividades ou fluxos, são os principais itens de configuração da MPS.

Funcionalidades comumente encontradas em ferramentas de gerência de configuração são (NAKAGAWA, 2006): (i) cria/remove repositórios; atualiza no repositório cada item de configuração alterado, acrescentando um item de configuração do repositório (*check-in*); (ii) retira um item de configuração do repositório e disponibiliza-o para o desenvolvedor (*check-*

out); (iii) compara configurações, ou itens de configuração, revelando as diferenças entre elas; (iv) cria e gerencia versões e *releases*; (v) e consulta configurações e outras informações persistidas como, histórico de acesso. Adicionalmente, um dos focos da gerência de configuração é o controle de mudanças. Para o processo, uma melhoria é uma mudança. Se as propostas de melhoria de processo estão registradas em uma ferramenta de rastreamento (Seção 5.6.3), será possível relacionar cada mudança específica à(s) proposta(s) de melhoria relacionada(s).

Para apoiar a documentação de processos, foi desenvolvida a solução Atabaque[47] (MALHEIROS et al., 2008b). A Atabaque apoia a definição/evolução de processos: (a) agilizando e minimizando a possibilidade de ocorrência de problemas de exibição e (b) possibilitando a exploração de diferentes formas de apresentação, potencialmente contribuindo para a usabilidade do processo.

A Atabaque é uma solução livre para documentar processos de forma colaborativa, baseada em padrões abertos como XML, XSL[48] e HTML. Sua principal vantagem é permitir que os criadores e mantenedores de processos se preocupem eminentemente com os problemas semânticos (problemas do processo diretamente relacionados com suas atividades, entradas, saída, artefatos, etc.) e não com os problemas de exibição. O isolamento das duas perspectivas aumenta a flexibilidade da evolução do processo. Encapsular alguns passos e gerar automaticamente alguns itens da documentação do processo aumentam a integridade do processo como um todo. Em sendo uma solução livre e multiplataforma a Atabaque pode ser estudada, alterada e evoluída por qualquer interessado.

Um processo definido/mantido em XML é facilmente lido por humanos, convertido para bancos de dados relacionais, transformado para exibição em HTML e integrado com ferramentas visuais de modelagem de processo. O fato de ser um padrão aberto facilita a manutenção colaborativa da Atabaque e viabiliza, por exemplo, que uma empresa compartilhe seu processo para que outras a tomem como modelo. São inúmeras as possibilidades de compartilhamento de conhecimento que são proporcionadas a partir da solução proposta.

A utilização de XML, e sua possível exibição em HTML, têm o benefício indireto de permitir a utilização de navegadores para consultar os processos, o que pode trazer ganhos em termos de usabilidade como:

---

[47] http://sourceforge.net/projects/atabaque/

[48] XSL - acrônimo para a expressão inglesa *Extensible Stylesheet Language*, que significa Linguagem de Folha de Estilo Extensível (BERGLUND, 2004).

- Facilidade de aprendizado: a interface do navegador é conhecida do grande público;
- Permanência: deve ser fácil lembrar-se de como navegar pelo processo, para que mesmo o usuário eventual possa retomar o seu uso após longos períodos sem utilizá-lo, sem a necessidade de aprender todos os procedimentos de novo; e
- Acessibilidade: flexibilidade para a sua utilização por pessoas com necessidades especiais, bem como a utilização em diferentes ambientes e situações, e por meio de vários equipamentos ou navegadores.

Com a Atabaque, a definição/evolução de processos pode ser descentralizada; executada por grupos específicos ou por usuários finais que tenham motivação pessoal e interesse na MPS. Nesse sentido, o uso da Atabaque pode ser conveniente para uma MPS distribuída e colaborativa.

A Atabaque está integrada ao CVS (e pode ser usada com outros sistemas de controle de versão), permitindo controle das alterações no processo. O conteúdo do processo armazenado em XML permite que o CVS apresente as diferenças entre duas versões de um mesmo arquivo, ou ainda que pessoas diferentes trabalhem em paralelo em áreas distintas de um mesmo XML (por exemplo, em sub-atividades de uma mesma atividade), efetivando a junção (*merge*) ao final. Ao se consolidar uma versão oficial do processo, também é permitido gerar linhas-de-base (*baselines*), recurso útil quando se deseja rastreabilidade das alterações.

Além da solução Atabaque, a solução livre EPF Composer (HAUMER, 2007), pode ser interessante para documentar processos. Essa ferramenta igualmente privilegia um formato amigável de documentação e disponibilização do processo. Diferentemente da Atabaque, que estrutura o processo de acordo com os elementos do RUP, a EPF Composer é compatível com o modelo SPEM (Seção 2.2.5). Pode ser mais difícil entendê-la e usá-la corretamente; adicionalmente, a EPF Composer não está pronta para ser usada por desenvolvedores que não tenham domínio da língua inglesa. No entanto, em sua versão mais recente, é uma solução mais completa que a Atabaque, permitindo, entre outras coisas, a geração gráfica dos fluxos do processo. As duas soluções foram analisadas em situações reais e ambas mostraram resultados positivos. Considerações sobre a aplicação de uma ou de outra solução estão disponíveis no Capítulo 6.

5.6.3 Estratégias para tratamento das propostas de melhoria de processo

O objetivo do tratamento das propostas de melhoria de processo é definir como são coletadas e encaminhadas propostas (solicitações/demandas/sugestões/) de melhoria de processo. Uma Proposta de Melhoria de Processo (PMP) é uma das entradas para a MPS (como foi ilustrado na Figura 37).

Ao utilizar o processo de desenvolvimento de software, os seus usuários (gerentes, desenvolvedores, testadores, etc.) identificam oportunidades de melhoria que podem tornar o processo mais simples, efetivo e adequado às necessidades desses usuários. Cada oportunidade de melhoria pode ser concretizada como uma PMP. Uma PMP pode ter como origem, dentre outras fontes (adaptado de SERPRO (2007)):

- Experiência do usuário com a utilização do processo;
- Recomendações geradas durante revisões conduzidas por grupos de garantia da qualidade;
- Ações de melhoria propostas em planos de ação, documentando o resultado de avaliações aplicadas no processo ou em unidades que o utilizam;
- Prospecção de novos métodos, técnicas e ferramentas para desenvolver software;
- Alinhamento do processo ao direcionamento tecnológico da empresa/comunidade; e
- *Benchmarking* com outros processos[49].

A identificação dessas oportunidades e a alteração contínua do processo para refleti-las podem ser mais efetivas se apoiadas por uma estratégia de gestão das PMPs que permita planejar, executar e acompanhar atividades de tratamento das propostas, definindo responsabilidades e provendo recursos para cada uma dessas atividades. Os benefícios de gerir sistematicamente as PMPs foram observados em um programa de MPS no SERPRO (Capítulo 4, Seção 4.2.3). À medida que os experimentos foram conduzidos os mecanismos para tratamento de PMPs da ColabSPI foram sendo evoluídos.

A partir da observação do tratamento de erros em projetos de software, em particular projetos de software livre, é possível estabelecer paralelo entre a gestão do tratamento de erros em software e o tratamento de PMP, pois ambos obedecem a um fluxo de tratamento que passa

[49] O *benchmarking* com outros processos pode ser interno ou externo e pode ser referente ao processo em si, ou às atividades necessárias para melhorar esse processo. O tema *benchmarking* em iniciativas de MPS é discutido por Zanetti et al. (2009). No contexto desta tese, o *benchmarking* será tratado apenas como mais uma fonte para a proposição de melhorias de processo de desenvolvimento de software.

por: submissão; avaliação; aprovação ou rejeição; implementação; e disponibilização. No âmbito do desenvolvimento de software, esse tipo de tratamento de erros vem sendo gerido por sistemas de *Bug tracking*[50] ou mais genericamente, sistemas de *issue tracking*.

A partir desse referencial e considerando a análise dos motivos de atraso no tratamento de PMP no SERPRO (Capítulo 4), a solução Mantis-PMP foi proposta (MALHEIROS et al., 2007b), englobando os mecanismos para tratamento de PMPs da ColabSPI.

A solução adotada foi inspirada em (MALHEIROS et al., 2007b):

1. **Concepção de desenvolvimento do software livre**. As características marcantes do desenvolvimento em software livre (REIS, 2003b) foram extrapoladas para a MPS. As seguintes características foram preservadas: (i) a gestão das PMPs é distribuída pela Internet; (ii) a melhoria é feita de maneira colaborativa e descentralizada; (iii) a participação na MPS é motivada por interesse pessoal do desenvolvedor, pois cada um contribui da forma que achar mais conveniente e interessados podem contribuir para a melhoria de processo;

2. **Experiência em MPS no SERPRO**. Em novembro de 2004, o SERPRO adotou uma forma sistematizada para tratar PMPs, por meio da solução GM-PSDS - Gestão de Mudanças no PSDS (Processo SERPRO de Desenvolvimento de Soluções) (MALHEIROS et al. 2006). A observação da GM-PSDS permitiu identificar os pontos fortes e fracos da solução e propor novas alternativas para gerenciar PMPs. Alguns experimentos foram conduzidos (Seção 6.2.2) e as lições aprendidas foram base para definir a nova solução proposta neste trabalho, Mantis-PMP, que foi adotada pelo SERPRO ;

3. **Tratamento de erros e sugestões por meio de ferramentas de *bug tracking***. Tal qual acontece nas manutenções evolutivas e adaptativas de software, é possível identificar e tratar fraquezas de uma versão de um processo de desenvolvimento de software, convertendo-as em melhorias para a próxima versão; e

4. **Práticas de melhoria de processos** disponíveis na literatura; como as práticas das áreas de processo do CMMI OPD (*Organizational Process Definition*) e OPF (*Organizational Process Focus*) (SEI, 2006).

A solução, batizada de Mantis-PMP por se tratar de uma adaptação do Mantis para o tratamento de PMPs, tem o objetivo de apoiar o SEPG corporativo na evolução e melhoria de processo de desenvolvimento de software. A solução viabiliza a análise, o controle e a monitoração de oportunidades de melhoria, vinculando-as aos ganhos obtidos para o processo; e

[50] *Bug tracking system* é uma aplicação de software projetada para ajudar à garantia de qualidade e aos programadores a monitorarem os erros reportados para o seu código.

estabelecendo um canal de comunicação entre o usuário do processo e o SEPG Corporativo para que o envio e tratamento de PMPs possam acontecer de forma sistemática.

Para a completa gestão das PMPs, a Mantis-PMP implementa o diagrama de estados apresentado na Figura 38. Esses estados são fundamentais para controlar o fluxo de dados disponibilizado pela solução e são resultantes do aprendizado obtido com a análise da MPS no SERPRO (Seção 6.2.2).

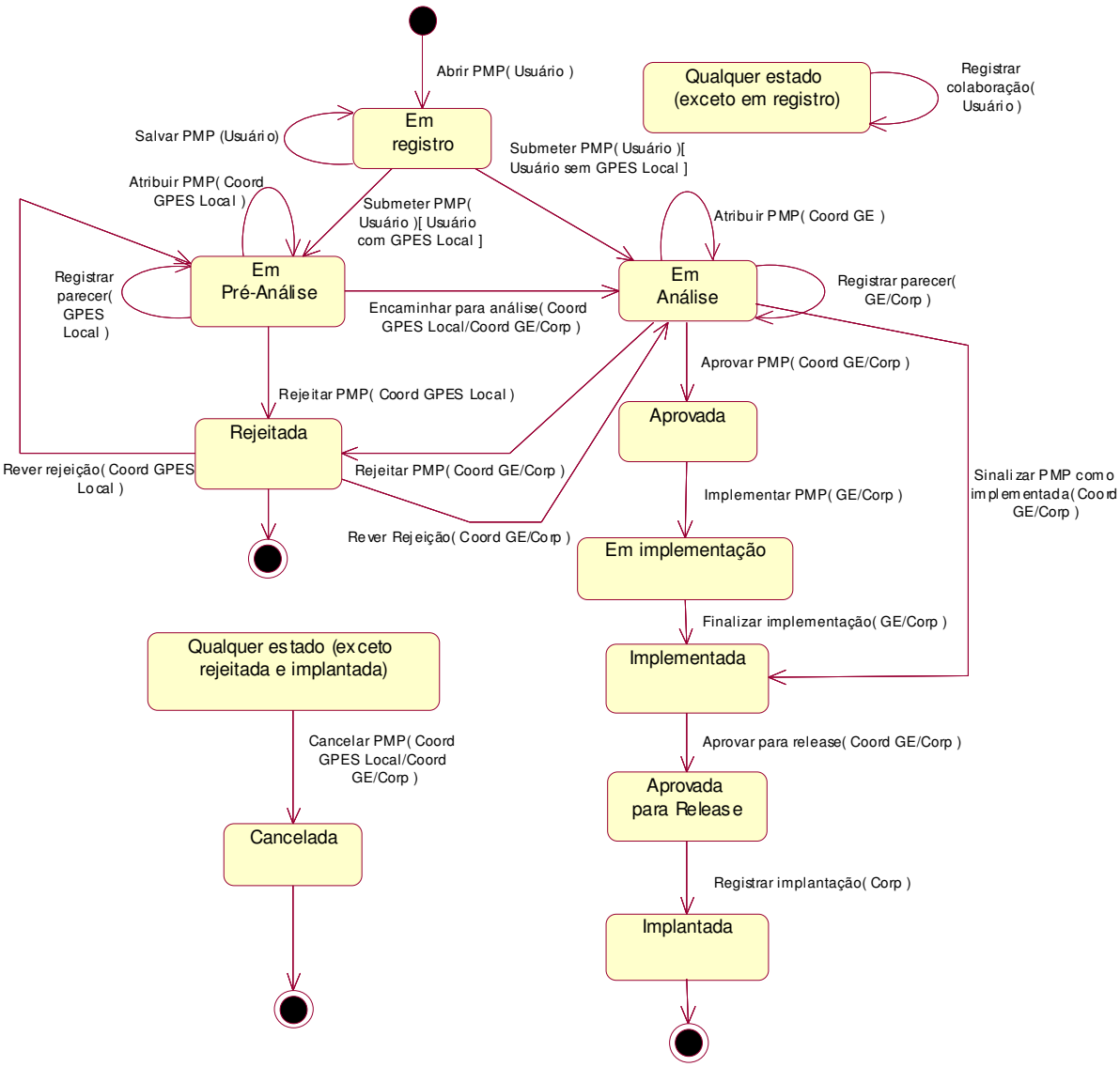


Figura 38: Diagrama de estado proposto para tratamento de PMP

Na Tabela 9, são apresentados os requisitos previstos para a Mantis-PMP. Para cada necessidade, é apresentada uma breve descrição do seu objetivo e a funcionalidade correspondente disponível na solução. Outras funcionalidades podem ser executadas em paralelo

ao fluxo da PMP para manipular os dados. Essas funcionalidades estão listadas na Tabela 10. Algumas dessas funcionalidades foram experimentadas, outras não. Quando não houve observação da funcionalidade em casos práticas, esse fato será indicado na tabela.

Uma PMP pode ser criada por qualquer um dos usuários do processo (ator **Usuário**). Enquanto está em elaboração, a PMP fica disponível apenas para este usuário que a criou no estado "Em registro". Nesse momento, a PMP é uma espécie de rascunho e pode ser totalmente alterada por seu criador. Nenhum outro usuário pode visualizá-la.

A partir do momento em que a PMP é submetida, tornando-se pública, ela fica disponível para análise de sua pertinência. Cada proposta deve ser analisada antes de ser incorporada ao processo de desenvolvimento de software. Essa avaliação será feita pelo SEPG corporativo. Na experiência do SERPRO, há evidências que essa decisão de aprovação pode também ser distribuída para os grupos especialistas, que podem auxiliar o SEPG corporativo aprovando PMPs menos polêmicas, de escopo restrito à sua área de atuação e de menor impacto para o processo, dando mais flexibilidade ao fluxo de tratamento de PMPs e aumentando a participação e a motivação dos grupos especialistas.

O **SEPG corporativo** é o ator responsável pela definição, manutenção e melhoria contínua do processo. Para avaliar as Propostas de Melhoria de processo, o SEPG corporativo pode precisar de um parecer especializado no domínio da proposta, que será emitido por um membro de um grupo **Especialista**.

Antes de serem analisadas pelo SEPG corporativo, as PMPs podem ser "pré-analisadas" pelo SEPG local, que atuará como um filtro de PMPs que devem ser rejeitadas. Essa atuação do SEPG local diminui a sobrecarga para o SEPG corporativo e permite uma primeira análise por um grupo que conhece mais de perto a necessidade local. Tal atuação só será possível quando o usuário que propôs a PMP for um desenvolvedor em uma das unidades distribuídas.

Propostas nitidamente não-procedentes, não-aplicáveis, não exeqüíveis com os recursos disponíveis, já registradas em outras propostas, ou que contrariem os objetivos de negócio da organização, devem ser "rejeitadas", indicando-se o motivo da rejeição. PMPs podem ser rejeitadas pelo SEPG local ou pelo SEPG corporativo. Caso não concorde com a rejeição o criador da PMP poderá solicitar revisão tanto para a instância que rejeitou a PMP, quanto para o SEPG corporativo. Propostas pertinentes e que tragam ganhos significativos ao processo devem ser "aprovadas". Uma vez aprovada, a PMP precisa ser "implementada" (ou seja, as alterações nos elementos do processo, como atividades e modelos de artefatos serão realizadas) para que de fato seja incorporada ao processo. Essa atividade será realizada pelo **desenvolvedor** e é gerenciada e acompanhada pelos grupos especialistas e pelo SEPG corporativo.

Tabela 9: Necessidades e funcionalidades previstas para a Mantis-PMP

Necessidade	**Descrição**	**Funcionalidade**
Criar PMP	A PMP é elaborada e gravada na base. Ela permanecerá "Em registro" como rascunho para o usuário criador até que seja submetida para análise dos grupos responsáveis. O usuário não precisa preencher uma PMP de uma única vez e poderá "Salvar a PMP" para complementar depois.	Abrir PMP Salvar PMP
Submeter PMP	A partir desse momento, a PMP é submetida para análise e fica visível a qualquer usuário interessado. Tanto o SEPG corporativo por analisar a PMP como o SEPCG local poderá pré-analisar, distribuindo o trabalho de análise e decisão entre o corporativo e o local.	Submeter PMP para análise
Solicitar Parecer	Caso seja necessário, o responsável por *analisar* a PMP no SEPG corporativo solicitará um parecer de um especialista, alocando a PMP para análise do mesmo.	Atribuir PMP
Emitir parecer	Possibilita ao especialista emitir a sua opinião e avaliação da PMP, bem como complementar a PMP, indicando o impacto para o processo.	Registrar parecer
Aprovar PMP	Caso o SEPG corporativo considere a PMP pertinente e avalie que trará ganhos para o processo, ele poderá aprová-la, para que ela seja incorporada a uma nova versão do processo. Na última versão de fluxo proposta para o tratamento de PMPs a aprovação pode ser realizada também por um grupo especialista. Essa possibilidade está em validação.	Atualizar estado para aprovada
Indicar que a PMP está em implementação	Depois que uma PMP é aprovada, alguma alteração precisará ser feita no processo, o que poderá envolver desenvolvimento de atividades, artefatos ou ferramentas. Enquanto a solução de uma PMP está sendo desenvolvida, a PMP estará "Em implementação"	Atualizar estado para: "Em implementação" (ver na Figura 38 "Implementar PMP")
Indicar que a PMP está implementada	Uma vez implementada uma PMP, seu estado deverá ser atualizado para implantada, indicando que a solução dada para atender a PMP está disponível para testes finais.	Atualizar estado para: "Implantada" (Figura 38, "Sinalizar a PMP como implementada" ou "Finalizar a implementação")
Disponibilizar PMP para ser incorporada	Quando uma PMP aprovada está pronta para entrar em produção em uma nova versão do processo, ela será marcada como "Aprovada para a release". Nesse caso a PMP já foi testada.	Aprovar para a *release*
Incorporar PMP	Quando uma PMP disponível para publicação é publicada em uma nova versão do processo, ela é considerada incorporada (implantada). Apenas PMPs aprovadas para *release* podem ser incorporadas ao processo.	Registrar implantação
Cancelar PMP	Uma PMP pode ser cancelada quando é decidido que ela não entrará no processo por um motivo diferente da rejeição, por exemplo: existe outra PMP semelhante que já está sendo tratada ou foi proposta outra solução que resolve a questão de forma mais efetiva.	Atualizar estado para "Cancelada"
Registrar colaboração	Essa funcionalidade tem o objetivo de aumentar a colaboração para tratamento de PMPs. Os usuários, de acordo com o seu interesse, podem implementar ou fornecer solução pronta que atenda a PMP, colaborando ativamente. Eles poderão também emitir opinião sobre os pareceres já registrados. Um caso particular de uso da funcionalidade é solicitar revisão de rejeição. Se o usuário proponente da PMP não concordar com a rejeição da mesma, ele pode solicitar nova análise de sua PMP rejeitada para o SEPG local ou corporativo.	Registrar colaboração
Rever rejeição	Caso o coordenador do GPES Local aceite a solicitação do usuário pode executar a funcionalidade "Rever rejeição" colocando novamente a PMP em pré-análise e pode encaminhá-la para análise do GE ou Corporativo e , nesse caso, a PMP segue seu fluxo normal.	Rever rejeição (retornar para o estado "em pré analise" ou "em análise")

Durante sua elaboração a PMP poderá passar por vários estágios, a saber: “Em implementação”, “Desenvolvida”, “Em teste”, “Testada”, até estar “Disponível” para implantação. O teste no contexto da MPS é a validação da nova versão do processo para identificar se, de fato, as mudanças aprovadas foram implementadas e se há ou não erros de conteúdo ou de exibição no processo. A partir desse ponto, a PMP estará disponível para implantação em uma próxima versão do processo. Por motivo de simplificação, escolheu-se representar esses estágios em apenas três estados: “Em implementação”; “implementada” e “Aprovada para release”. Uma vez disponibilizada em uma versão do processo, a PMP será considerada “Implantada”.

Tabela 10: Demais funcionalidades do Mantis-PMP.

Funcionalidade	**Descrição**
Copiar PMP	Gerar a cópia de uma PMP, alterando apenas o número de identificação. Essa funcionalidade pode agilizar a criação de novas PMP que tenham conteúdo parecido com PMP já existente.
Atribuir PMP	Permitir indicar um usuário que será o responsável pelo tratamento daquela PMP até uma nova ação que transfira esta responsabilidade, sendo todas estas alterações gravadas em Histórico de Eventos.
Atualizar PMP	Permitir atualizar os diversos campos da PMP. Os campos básicos como descrição, resumo ou anotações estão disponíveis para atualização por outros usuários, sendo a alteração notificada por mensagem de correio eletrônico ao autor da PMP. A atualização de campos personalizados (ver item 2.4.1.2) poderá ser realizada a depender do perfil do usuário, de acordo com configurações específicas. Essa funcionalidade não foi experimentada em um caso prático.
Atualizar versão alvo	Permitir programar/ reprogramar publicações de versões por meio da atualização em lote da versão alvo das várias PMP selecionadas.
Atualizar prioridade	Permitir planejar o tratamento de PMP por meio da indicação das propostas mais prioritárias. Essa atualização pode acontecer em lote, a critério do usuário. O uso de prioridades pode contribuir para uma gestão mais efetiva das mudanças no processo.
Relacionar PMP	Permitir estabelecer relacionamento entre duas PMP para minimizar o trabalho de tratamento das PMPs que estão relacionadas umas com as outras. Também pode ser usado para indicar duplicidades.
Consultar Histórico de Eventos	Permitir consultar histórico de eventos. Todas as atualizações citadas são registradas em um histórico e disponibilizadas numa consulta simples, que descreve todas as mudanças realizadas, quando e por quem.
Consultar PMP por estado	Permitir consultar as PMP pelo estado em que ela se encontra. Esta consulta servirá de insumo para identificar pontos de retenção no andamento das PMPs.
Consultar PMP por usuário	Permitir consultar as PMP sob responsabilidade dos usuários que evoluem o processo; sejam eles membros dos grupos especialistas ou desenvolvedores.
Consultar PMP por versão do Processo	Permitir consultar o histórico de cada uma das versões liberadas para o processo.
Controlar acesso	Permitir controle de usuários cadastrados na Mantis-PMP, indicando seu papel para a MPS
Anexar documentos	Permitir anexar documentos à PMP em qualquer estado, à exceção dos estados finais.
Notificar mudanças de estado	Emitir notificações padrão por e-mail que alertem aos usuários sobre o andamento do tratamento da PMP. Exemplos de notificação propostos por um membro do SEPG corporativo no SERPRO estão disponíveis no Apêndice A.

"Implantada" ou "Cancelada" são estados finais do fluxo de tratamento das PMPs. Estando em um desses estados, a PMP não mais poderá sofrer alteração. Os usuários podem registrar sua opinião sobre qualquer PMP submetida, exceto quando a PMP estiver no estado "Em registro".

5.6.4 Colaboração e comunicação

O módulo de colaboração e comunicação deve contribuir para a MPS, permitindo acesso e manipulação das suas informações e promovendo um ambiente cooperativo.

Os mecanismos de comunicação apresentados por Booch e Brown (2002) foram considerados para incorporação na ColabSPI. Foram selecionados:

- **Listas de mensagens de correio eletrônico** – correio eletrônico é um mecanismo de comunicação difundido, que permite comunicação assíncrona entre os usuários. O uso de listas de mensagens de correio eletrônico será aplicado para grupos pequenos com objetivos em comum, comunidades que estejam começando e anúncios. Deve ser permitido aos usuários solicitar inscrições em listas de seu interesse. Listas também são úteis para mensagens de congratulação e as mensagens de correio devem ter um link para o site do processo (YEAKLEY e FIEBRICH, 2007);
- **Últimas notícias** – deve ser utilizado como mecanismo para divulgar novidades e comunicar eventos. Por exemplo, o SEPG corporativo pode utilizar esse mecanismo para informar o lançamento de uma nova versão do processo ou divulgar uma conferência sobre MPS. Pode ser interessante oferecer o recurso de "*feeds*" [51] ao usuário. O "*feed*" permite aos usuários se inscreverem em um site e receberem atualizações, permanecendo informado; e
- **Fórum** – deve ser utilizado para discussões de questões mais amplas que as questões discutidas nas listas de mensagens de correio eletrônico. Yeakley e Fiebrich (2007) recomendam que eles sejam abertos para os usuários e divertidos.

Na ColabSPI, mecanismos de comunicação, e alguns de colaboração, podem ser providos pela adaptação de um ADC (MALHEIROS et al., 2009c).

O ADC (Seção 2.4) é um espaço virtual, onde todas as partes interessadas em um projeto – até mesmo aqueles que trabalham em diferentes locais ou momentos - podem negociar, discutir, compartilhar conhecimento e dividir algumas tarefas para criar entregas e artefatos. O ADC provê acesso integrado aos diferentes mecanismos e ferramentas, criando um espaço

[51] http://pt.wikipedia.org/wiki/Feed

virtual para um projeto específico. Em sendo esse projeto a evolução de um processo, é possível imaginar um espaço onde membros do SEPG, grupos especialistas e interessados na MPS possam trabalhar juntos na melhoria deste processo. E, da mesma maneira, negociar, discutir, compartilhar conhecimento e dividir algumas tarefas; agora, relacionadas à MPS e não ao desenvolvimento de um software.

Os ADC estão sendo incorporados ao dia-a-dia dos desenvolvedores de software paulatinamente, em especial no desenvolvimento de software livre. A maioria dos grandes projetos em software livre está disponibilizada em um ADC (ex.: SourceForge). Além disso, estudos apresentam iniciativas de utilizar ADC no desenvolvimento de software corporativo (Seção 2.5.4). Assim, utilizar um ADC para a MPS, para fornecer mecanismos de comunicação, coordenação e colaboração, pode ter o benefício de prover um ambiente que já é familiar aos desenvolvedores e, por esse motivo, estimular e facilitar o seu uso.

Diversos ADC estão disponíveis e com adaptações pequenas podem apoiar a MPS.

Na Figura 39, é ilustrada uma simulação do ambiente de comunicação da ColabSPI, de um projeto de MPS criado em um ADC. A tela apresenta o ambiente de MPS do processo "ColabProcess". Esse ambiente pode ser acessado por uma única URL, indicação "1", na figura.

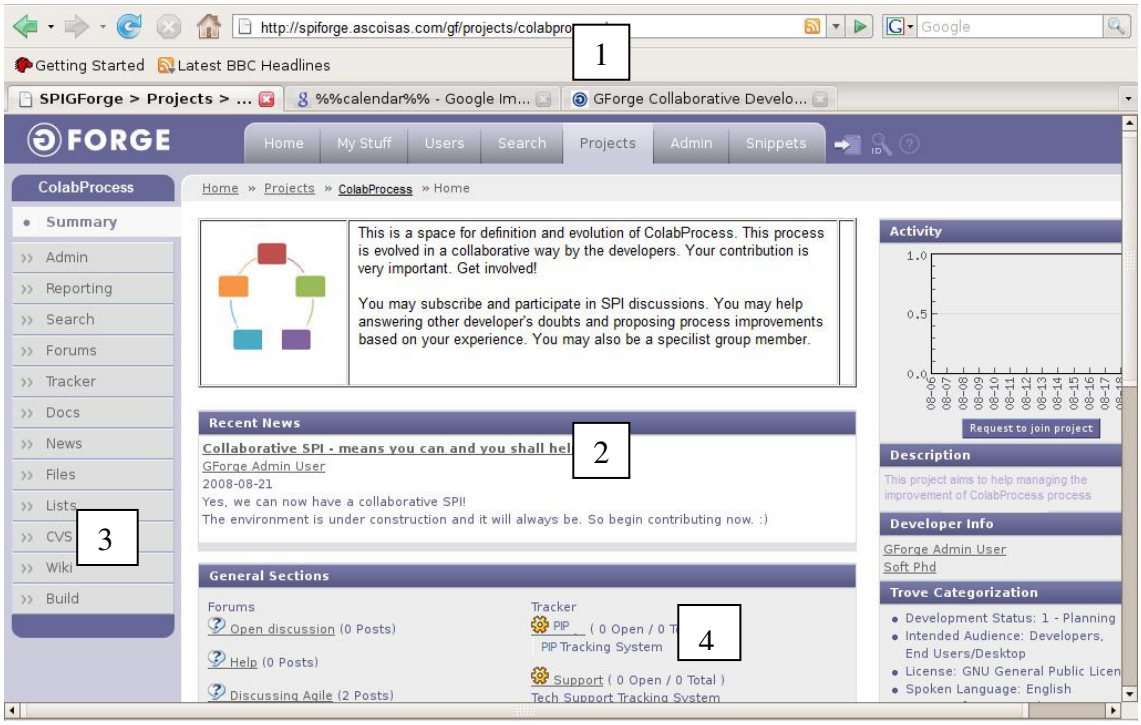


Figura 39: O espaço virtual para o projeto de MPS

Uma vez no espaço virtual do projeto da MPS, todas as atividades de comunicação, documentação e tratamento de PMPs estão disponíveis: fóruns, listas de mensagens de correio eletrônico, dados históricos, fluxos de PMPs, etc. (Figura 39, identificação "3"). Na página

principal, é possível saber as últimas notícias relacionadas à MPS (identificação “2”, Figura 39). Além disso, informações relacionadas aos atores da MPS também são disponibilizadas aos usuários pelo ADC. Ainda, a partir da página principal (Figura 39, identificação “4”), é possível acessar as atividades relacionadas ao tratamento de PMPs e todas as funcionalidades apresentadas na Seção 5.6.3.

O ADC provê acesso também a um ambiente Wiki, onde as PMP podem ser discutidas. Trechos do processo relacionados a uma PMP, que precisam de mais discussões, podem ser copiados para o ambiente wiki e ser editados pelos colaboradores da MPS até que adquiram um formato adequado para incorporação do processo.

5.7 Considerações finais

Neste capítulo, estratégias e mecanismos para apoiar uma MPS colaborativa e distribuída foram discutidos e organizados em uma estratégia denominada ColabSPI. Serviram como base para estabelecer os elementos da ColabSPI os fatores críticos de sucesso da MPS, em particular os fatores ligados a aspectos de: comunicação; colaboração, inclusive aspectos de participação e motivação dos desenvolvedores; evolução e manutenção de processos; e gestão da implementação da MPS. A ColabSPI é compatível com boa parte dos modelos e guias de MPS por permitir a evolução dos processos de forma iterativa, além de permitir a implementação das boas práticas sugeridas, por exemplo, no CMMI. Uma referência importante para estruturação da ColabSPI foi o modelo de arquitetura de GC de Lindvall et al. (2002), adaptado ao contexto da MPS, sobretudo, por permitir a visualização dos diferentes níveis de abstração dos mecanismos de apoio à GC que poderiam ser adaptados para a ColabSPI.

A estratégia ColabSPI foi dividida em quatro elementos: (i) Princípios, (ii) Estrutura organizacional para MPS; (iii) passos para a MPS; e (iv) infraestrutura de MPS. Cada um destes elementos foi detalhado a partir das seguintes influências: (i) aspectos do DDS; (ii) práticas adotadas no paradigma do desenvolvimento de SL; (iii) práticas de GC; (iv) ferramentas de software que disponibilizam serviços colaborativos, como abordagens Wiki; e (v) sistema para acompanhamento do tratamento de erros em software (*Bug tracking system*).

Para compor o elemento infraestrutura da ColabSPI, procedimentos, práticas e ferramentas, já aplicados ao desenvolvimento de software, foram considerados; sempre sob a perspectiva da evolução e do uso do processo. Dentre esses, destacam-se: (i) mecanismos para controle de versão; (ii) ferramentas para tratamento de erros em software; e (iii) mecanismos para comunicação e coordenação de atividades. A adaptação desses procedimentos, práticas e

mecanismos para a MPS foi discutida. A infraestrutura foi dividida em três grupos funcionais que nortearam os estudos experimentais conduzidos para sua validação (Capítulo 6): (i) *documentação e evolução do processo em si;* (ii) *tratamento de PMP;* e (iii) *comunicação e colaboração*.

Além do estabelecimento dos mecanismos da infraestrutura da ColabSPI, que indicam como boas práticas de MPS podem ser aplicadas, implementações de tais mecanismos foram discutidas. Para a *documentação e evolução do processo em si,* foi definida e desenvolvida a solução Atabaque, uma solução livre para documentação de processos. Ademais, foi considerado o uso da solução EPF Composer. Para o *tratamento de PMPs*, um fluxo de trabalho e as principais atividades foram definidos e uma adaptação da solução Mantis foi proposta, dando origem à solução Mantis-PMP. Por fim, para *comunicação e colaboração*, a aplicação dos atributos e das funcionalidades de um ADC foi discutida.

No próximo capítulo, aplicações práticas das estratégias e mecanismos da ColabSPI são apresentadas em diferentes estudos experimentais em uma organização de grande porte. Além disso, a aplicação da ColabSPI para a evolução de um modelo de maturidade é ilustrada, por meio de uma proposta de adaptação para o OMM.

6- Estudos de caso de aplicação da ColabSPI

Neste capítulo, apresentam-se os resultados de estudos experimentais relacionados à estratégia e infraestrutura ColabSPI descritas no Capítulo 5. Antes de detalhar tais estudos, é apresentada uma contextualização dos tipos de estudos experimentais utilizados para proporcionar evidências sobre a qualidade e a utilidade das estratégias para MPS propostas (Seção 6.1).

Os estudos experimentais conduzidos foram agrupados em duas seções. Na Seção 6.2, são apresentados estudos e observações da MPS na indústria. Foram conduzidos estudos, em diferentes níveis de informação, para analisar a estratégia ColabSPI e seus mecanismos relativos a: (i) documentação e manutenção de processos; (ii) tratamento de PMPs; e (iii) comunicação e colaboração. Além disso, foram conduzidos estudos relacionados ao uso de projeto piloto como instrumento para a MPS. Na Seção 6.3, são compiladas observações da aplicação da estratégia ColabSPI para a evolução do modelo de maturidade OMM.

6.1 Tipos de estudos experimentais utilizados

A transferência tecnológica da academia para a indústria não é uma atividade trivial e pode ser demorada (PFLEEGER, 1999). Vários fatores (como orçamento, prazos, disponibilidade de recursos, retorno de investimento e, até, a sua utilidade prática) podem influenciar a viabilidade de aplicação de uma solução na prática, às vezes, tornando uma idéia teoricamente consistente inviável em determinado contexto industrial. Segundo Shull et al. (2001), a influência desses fatores, muitas vezes, não pode ser analisada em laboratórios. Quando uma nova tecnologia (ou, em particular, um novo processo) não demonstra uma melhoria significativa no contexto industrial, é difícil identificar a real causa do insucesso, que poderia ser, entre outras: (i) o processo original tem falhas; (ii) o processo não se encaixou no contexto empresarial específico; ou (iii) o processo estava correto, mas sua aplicação não foi adequada (SHULL et al., 2001).

Para converter melhorias de processos conceituais em melhorias aplicadas na indústria, Shull et al. (2001) propõem uma metodologia experimental iterativa, que apresenta uma série de questões e diferentes tipos de estudos que podem auxiliar na resposta a essas questões. Tal metodologia foi utilizada como referência para conduzir os estudos experimentais utilizados nesta tese, e cujas execuções estiveram sujeitas, não só a requisitos metodológicos, como também a requisitos ditados pelo ambiente industrial, que impõem, por si só, desafios específicos. Diferentemente da metodologia proposta por Shull et al.(2001), os estudos aqui apresentados, via de regra, não utilizam estudantes como participantes e, sim, profissionais. Ainda assim, a idéia de utilizar diferentes tipos de estudos em uma escala crescente de informações observadas foi muito útil e tomada como referência.

Com essa perspectiva, foram considerados os tipos de estudos experimentais descritos por Wohlin et al. (1999). De acordo com eles, tais tipos incluem: (i) pesquisa de opinião (*surveys*); (ii) estudos de caso e (iii) experimentos controlados. Para analisar as estratégias propostas, foram utilizados estudos de caso. Estudos de caso são conduzidos com o propósito de investigar uma entidade ou fenômeno dentro de um espaço de tempo específico. São utilizados principalmente para monitorar atributos presentes em projetos, atividades ou atribuições (MAFRA e TRAVASSOS, 2006). Dados são coletados de acordo com um objetivo específico do projeto. Um determinado atributo é monitorado (por exemplo, confiabilidade ou custo) e dados são coletados com o propósito de medir este atributo[52].

Kitchenham et al. (1995) defendem que estudos de caso auxiliam a indústria a avaliar os benefícios de métodos e ferramentas, além de ser uma forma econômica e efetiva para assegurar que uma mudança no processo propiciará os resultados desejados. Estudos de caso podem ser organizados de diferentes formas. Por exemplo, podem-se comparar os resultados ao usar métodos novos com os dados de referência existentes em uma organização; podem-se comparar resultados de projetos similares, quando um deles utiliza uma nova tecnologia, e o outro utiliza uma tecnologia corrente; ou, pode-se aplicar uma solução aleatoriamente a um projeto da organização e não aplicar aos demais.

No que tange a forma de coleta de dados para a execução dos estudos, segundo Travassos et al.(2002), existem três métodos principais: (i) histórico; (ii) de observação; e (iii) controlado. O método histórico é utilizado para coletar dados experimentais dos projetos que já tenham sido terminados. Os dados já existem e é preciso analisá-los. O método de observação coleta dados relevantes enquanto o projeto está sendo executado. O método controlado também coleta dados

[52] http://ese.cos.ufrj.br/wikiese

enquanto o projeto está sendo executado, mas provê instâncias múltiplas de uma observação oferecendo validade estatística dos resultados do estudo. Para os estudos aqui apresentados foram utilizados dados históricos e dados de observação.

Basili (1996) propõe que os objetivos para os estudos em engenharia de software sejam estabelecidos de acordo com os parâmetros propostos para as metas do modelo GQM (BASILI et al., 1994a). Esse foi o formato adotado para os estudos desta tese.

6.2 Aplicação da ColabSPI para melhoria de processos na indústria

Estudos experimentais foram conduzidos para analisar a aplicação da estratégia ColabSPI para a MPS na indústria (seções 6.2.1 a 6.2.3). Os estudos foram organizados em três conjuntos, que permitem analisar diferentes grupos funcionais da ColabSPI:

- a documentação e evolução estruturada de processos, com o uso da Atabaque e da EPF Composer como soluções de apoio (Seção 6.2.1);
- o tratamento de PMPs, por meio de um fluxo de trabalho, apoiado pelas soluções GM-PSDS e Mantis-PMP (Seção 6.2.2); e
- o uso de mecanismos de comunicação, por meio do uso de um ADC para a MPS (Seção 6.2.3).

Cada conjunto corresponde a um grupo funcional da infraestrutura da ColabSPI (Figura 35). Para cada conjunto, é apresentada uma visão integrada de seus estudos com os principais objetivos de cada estudo e a sua evolução. Para cada estudo são descritos: objetivo, participantes, material para estudo, resultado e lições aprendidas.

Além dos estudos relacionados a cada grupo funcional da arquitetura proposta, foram conduzidos estudos que permitiram verificar a adequação do uso de projetos piloto para introdução de melhorias em processos de desenvolvimento de software (Seção 6.2.4).

6.2.1 Estudos relacionados à documentação e à evolução estruturada de processos

Para analisar os mecanismos relacionados à documentação e à evolução de processos, três cenários foram considerados: evolução do processo sem o apoio das estratégias previstas na ColabSPI; o uso da solução Atabaque e o uso da solução EPF Composer.

Na solução Atabaque, descrita na Seção 5.6.2, mecanismos para autoria de processos permitem a editoração de versões do processo. Além disso, para acessar o resultado final da

documentação do processo (publicada no formato HTML em um servidor HTTP) é necessário um navegador web.

A solução Atabaque é baseada em componentes, permitindo a aplicação das suas funcionalidades como um todo ou o uso de apenas alguns deles. As principais funcionalidades disponíveis na Atabaque são: *Componente Editor Visual; Componente Gerador de Site* e *Processo Modelo*. Cada uma dessas funcionalidades é detalhada em Malheiros et al. (2008). Para os estudos experimentais apresentados nesta seção, os três componentes da Atabaque foram utilizados.

A divulgação da solução permitiu que os autores tivessem contato com a abordagem de desenvolvimento colaborativo do software livre. Interessados na solução apareceram em alguns lugares do país, um deles tornando-se desenvolvedor ativo da Atabaque. Em função de comentários recebidos, o editor visual da Atabaque foi melhorado e uma nova versão foi disponibilizada. O interesse na Atabaque pode ser demonstrado pelo histórico de *download*: entre a Atabaque e sua documentação foram mais de 1.800 *downloads* (Figura 40) desde sua disponibilização no Sourceforge e até o final de agosto de 2009. É importante ressalvar que o histórico de *download* não demonstra satisfação dos usuários ou o uso efetivo da solução, apesar de demonstrar interesse no assunto.

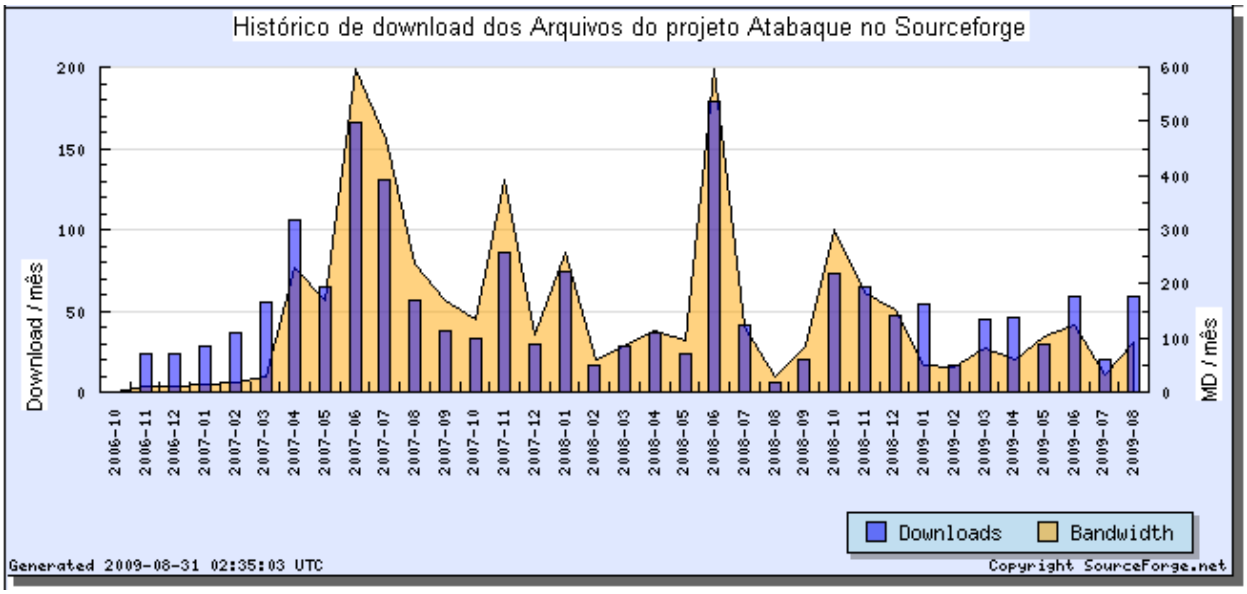


Figura 40: Histórico de *download* para os arquivos da Atabaque

A seqüência dos estudos relacionados à documentação de processos e uma síntese da evolução dos resultados são ilustrados na Figura 41. Os três estudos (Estudo de viabilidade; Estudo de Caso 1 e Estudo de Caso 2) são detalhados a seguir.

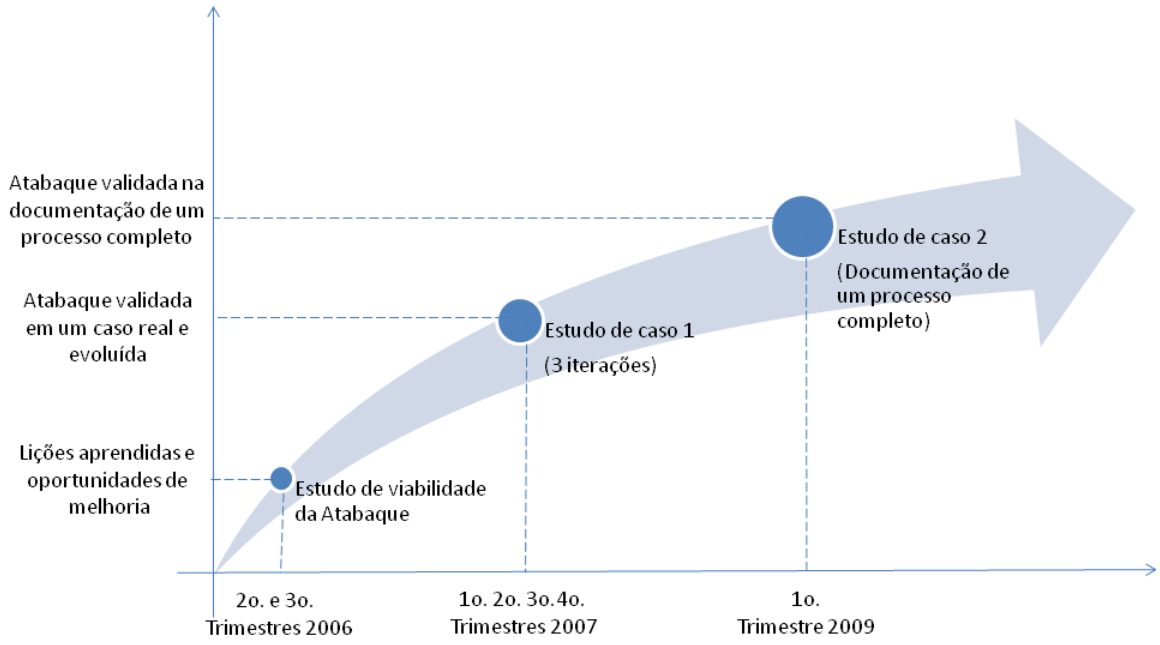


Figura 41: Evolução dos estudos sobre documentação estruturada de processos

Estudo de viabilidade da Atabaque

O objetivo do estudo de viabilidade foi identificar se o uso da Atabaque poderia ser adequado para documentar e evoluir um processo de software.

Objetivo: ***Analisar*** o uso da Atabaque para documentar processos ***com o propósito de*** caracterizá-la ***quanto a*** sua eficiência e a qualidade do processo resultante, ***do ponto de vista*** do documentador do processo, ***no contexto*** de uma melhoria distribuída de processos em uma organização de grande porte.

Hipótese: A utilização de uma solução que permita a autoria de processos de forma estruturada e a separação do conteúdo e do formato de apresentação do processo podem trazer benefícios para a documentação de processos com respeito à qualidade do processo final e à eficiência da evolução. Para esse estudo, entende-se por eficiência o tempo necessário para publicar uma versão do processo, uma vez definido o conteúdo dessa versão. A qualidade do processo resultante para esse estudo está relacionada à quantidade de erros de exibição e às inconsistências no relacionamento entre os componentes do processo (atividades, papéis, etc.).

Participantes: Dois participantes executaram o estudo de viabilidade da Atabaque. Ambos possuíam experiência industrial e eram desenvolvedores do Atabaque, sendo que um participante (a autora desta tese) atuava como membro do SEPG corporativo e outro como gerente sênior de uma das equipes de desenvolvimento de software.

O estudo de viabilidade foi conduzido de forma distribuída, estando um participante no estado de São Paulo e o outro na Bahia.

Material para estudo: Dado que o objetivo, nessa etapa, era avaliar preliminarmente a viabilidade de usar a Atabaque para documentar processos, foi utilizado um processo simplificado fictício que contém os principais elementos de um processo de desenvolvimento de software (atividades, papéis, etc.). O processo fictício foi denominado de PADS - Processo Atabaque de Desenvolvimento de Soluções e está disponível no *site* da Atabaque. No PADS, uma *Macroatividade* (correspondente a "disciplina" do RUP (IBM, 2010)) é composta de uma ou mais *Atividades* que, por sua vez, possuem uma ou mais *Sub-atividades* associadas. Cada Sub-atividade possui um *Responsável*, um ou mais *Participantes*, uma ou mais *Entradas* e uma ou mais *Saídas*. Entradas e Saídas são *Artefatos*. Responsável e Participantes são *Papéis*. Macroatividade, Atividade, Sub-atividade e Papéis possuem sempre um identificador, um nome e uma definição. Artefatos podem apontar para um modelo (*template*) de documento. A estrutura do PADS está disponível em Malheiros et al. (2008).Vale ressaltar que o PADS não é um processo completo. Ele foi criado apenas para apoiar o estudo de viabilidade. No entanto, é possível reutilizar o seu formato de publicação, considerando que a Atabaque separa o conteúdo do processo do seu formato.

Resultado e Lições Aprendidas

Na Figura 42, é mostrado o editor visual da versão da Atabaque utilizada para o estudo de viabilidade. Nesta figura, apresenta-se como seria a inclusão dos artefatos cronograma(CR), plano de projeto (PP) e relatório de acompanhamento do projeto (RAP) como entradas para a subatividade "Verificar andamento do projeto".

Na Figura 43, é mostrado o **resultado da geração do processo** fictício utilizando a Atabaque. É possível observar "PP", "CR" e "RAP" como *Entradas* para a sub-atividade "Verificar andamento do projeto". O processo foi publicado no formato da Figura 43 a partir das informações de conteúdo informadas no editor visual (Figura 42).

A Atabaque mostrou-se um instrumento viável para a geração de um processo e manipulação de seu conteúdo. O uso da Atabaque pode trazer benefícios para a documentação de processos, entre eles:

a) Rápida geração de páginas HTML, uma vez definidos os arquivos XSL e XML (eficiência);
b) Reutilização da estrutura das páginas (XSL), proporcionando agilidade na geração de novas páginas (eficiência) e minimizando problemas de inconsistência no formato das páginas (qualidade do processo);

c) Facilidade de alterar o formato padrão de elemento do processo, exigindo a manutenção de apenas um arquivo XSL, ao invés de vários arquivos HTML que incluiriam não apenas o formato, mas também o conteúdo do processo (qualidade do processo); e

d) Provável redução de erros de exibição (tamanho, tipo e cor de texto, *links* e tabelas) em função de todas as características de formatação estarem concentradas em um único XSL (qualidade do processo).

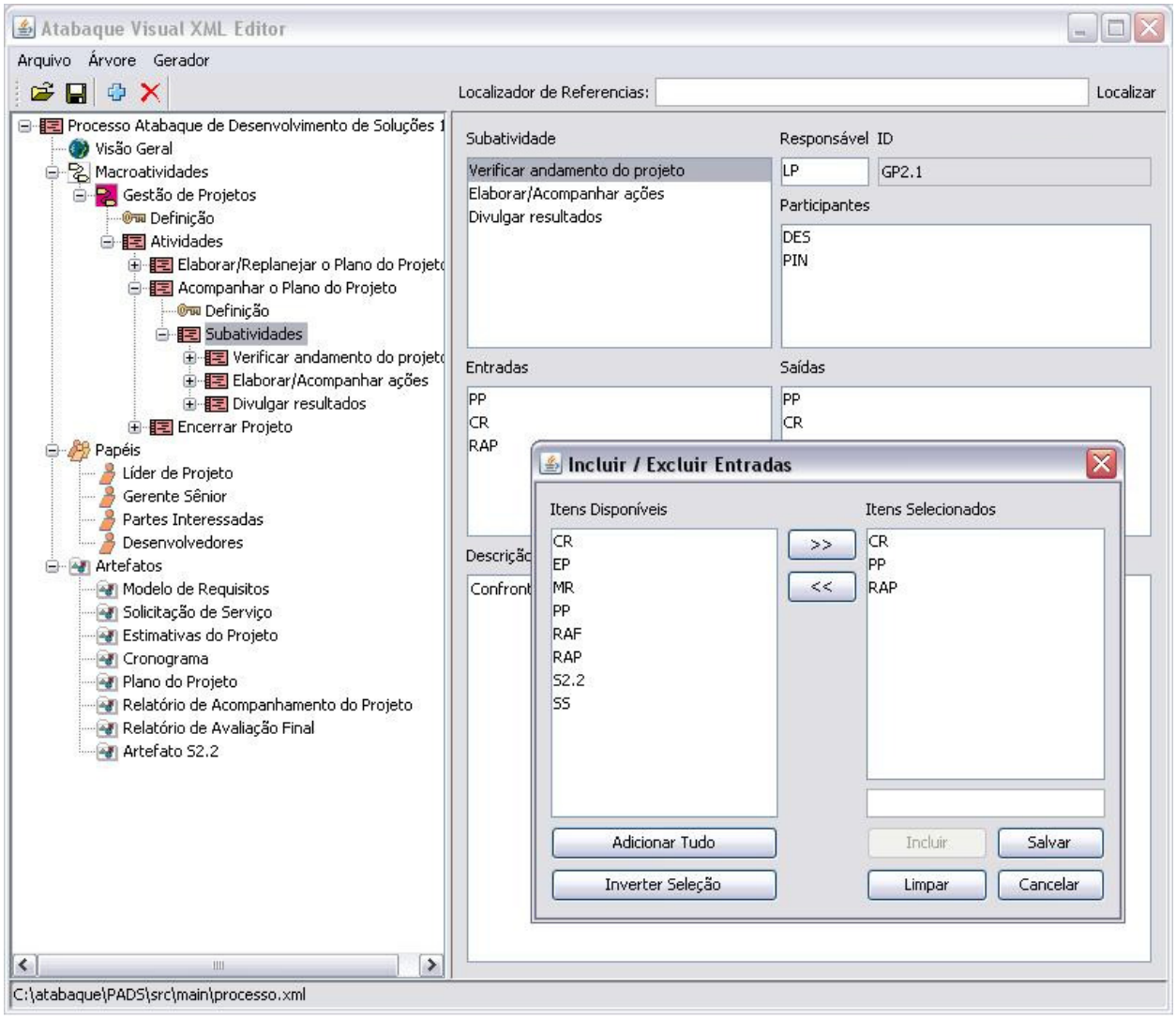


Figura 42: Editor visual - alterando a macroatividade “Gestão de Projetos” do PADS

A existência de telas gráficas para editar o conteúdo do processo é uma das principais vantagens da Atabaque. O uso de telas gráficas traz a possibilidade de pessoas com um baixo conhecimento na tecnologia web manterem o *site* do processo. Foi identificado que o editor visual precisaria ser melhorado para permitir a inclusão de recursos de edição de textos (negrito, itálico, *links*, etc.) e a implementação de todos os elementos básicos de um processo.

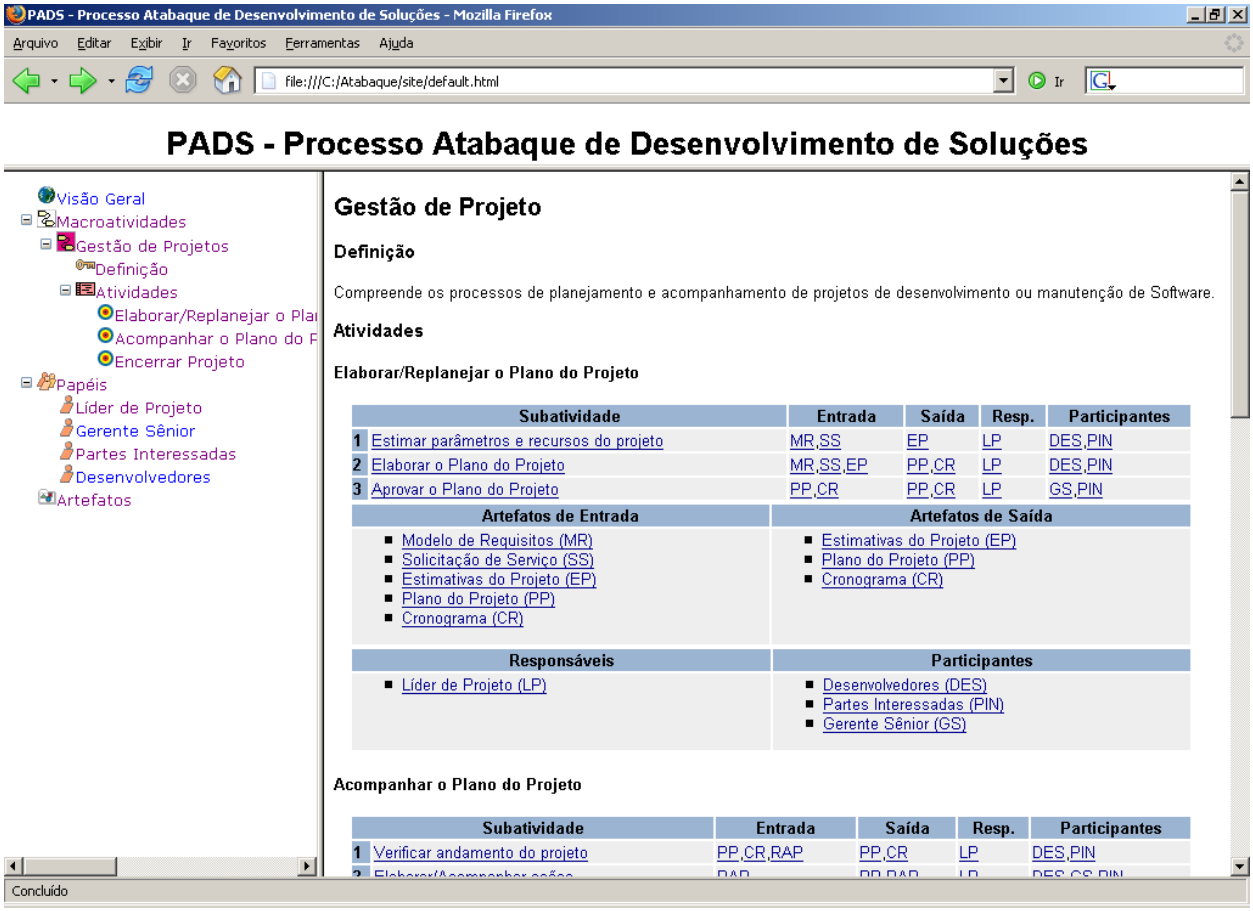


Figura 43: *Site* do PADS publicado durante o estudo de viabilidade

Estudo de Caso 1

Após a sua criação e o estudo de sua viabilidade, a Atabaque foi aplicada em um estudo de caso, para validar e identificar oportunidades de melhoria. A Atabaque foi utilizada em um ciclo real de MPS. Neste estudo, realizado durante o ano de 2007, ocorreram três iterações. Os seus resultados foram apresentados em Malheiros et al. (2008). A Atabaque foi adotada para evoluir uma das macroatividades do processo padrão de uma organização de desenvolvimento de software de grande porte. Na época do estudo, a MPS para esse processo já acontecia de forma distribuída, estando os membros do SEPG corporativo distribuídos em quatro cidades e os membros dos grupos especialistas, cerca de 100, em dez cidades.

Objetivo: O objetivo do estudo de caso foi ***analisar*** o uso da Atabaque em uma aplicação real para ***avaliá-la quanto a*** sua eficiência e eficácia ***do ponto de vista*** do documentador do processo ***no contexto*** da evolução de parte de um processo de software em uma organização de grande porte.

Hipótese 1 (H1): O uso da Atabaque para apoiar a evolução de um processo de software pode melhorar a eficácia da documentação do processo, diminuindo a ocorrência de erros.

Hipótese 2 (H2): O uso da Atabaque para apoiar a evolução de um processo de software pode melhorar a eficiência da documentação do processo, diminuindo o esforço para manter o processo.

Participantes: Três membros de um Grupo Especialista de uma macroatividade de Gestão de Projetos de Software participaram do estudo de caso. Na época do estudo, a autora desta tese atuava como supervisora do SEPG corporativo dos trabalhos de Gestão de Projetos de Software e foi uma das participantes do estudo. Os três membros possuíam experiência na evolução do processo documentado em formato HTML.

Material para estudo: Novas versões do processo de Gestão de Projetos de Software geradas a partir da Atabaque. Os conteúdos das novas versões representavam implementações de evoluções do processo a partir de PMPs reais. Para este estudo de caso dados foram coletados e os ciclos de melhoria para publicação das versões: 6.3, 6.4 e 6.5 do processo foram observados.

Resultados e Lições Aprendidas

Três publicações da macroatividade de Gestão de Projetos com o uso da Atabaque foram monitoradas. O sistema de gerenciamento de versões CVS foi utilizado para armazenar as versões do processo. As PMPs geradas posteriormente para cada versão foram monitoradas e avaliadas.

Após o uso da Atabaque, o número de PMPs relacionadas a questões/erros de exibição reduziu bastante, indicando que a Atabaque pode contribuir para a eficácia da documentação do processo. Nas duas últimas versões do processo, publicadas imediatamente **antes** do uso **da Atabaque,** 13% das propostas de melhoria eram relacionadas a questões de exibição. Nas duas primeiras versões do processo publicadas **com o uso da Atabaque**, esse percentual caiu para 2%. Na última publicação do processo que foi monitorada no escopo do Estudo de Caso 1, nenhum erro de exibição foi reportado, sendo que no encerramento do estudo de caso, o processo estava disponível há mais de dois meses.

Em uma das iterações, a Atabaque foi avaliada no contexto de uma manutenção de grande porte da disciplina Gestão de Projetos. Os participantes fizeram uma evolução no processo que envolveu: eliminar 5 artefatos, 4 atividades e 55 sub-atividades; alterar 30 sub-atividades e 7 atividades e a criar 1 atividade. A manutenção total de 100% das atividades e 83% das sub-atividades foi feita em apenas 5 dias úteis (MALHEIROS et al. 2008). Uma manutenção equivalente, sem o uso da Atabaque, demandaria ao menos 10 dias úteis, ou 100% a mais de esforço, com base no histórico registrado de esforço de manutenções para o processo e a

comparação do escopo da manutenção com manutenções anteriores. Ou seja, o uso da Atabaque pode aumentar a eficiência da documentação de processos.

Estudo de Caso 2

Em função dos resultados positivos do uso da Atabaque no processo de desenvolvimento de software, a solução foi escolhida para documentar/evoluir o processo de Gestão de Projetos no SERPRO. Mais uma vez a utilização inicial da Atabaque, no novo contexto, foi monitorada.

Objetivo: O objetivo do estudo de caso foi ***analisar*** o uso da Atabaque em uma aplicação real para ***avaliá-la quanto a*** sua eficiência ***do ponto de vista*** do documentador do processo ***no contexto*** da evolução de um processo completo em uma organização de grande porte.

Hipótese 1 (H1): O uso da Atabaque para apoiar a evolução de um processo pode melhorar a eficiência da documentação do processo, diminuindo o esforço para manter o processo.

Participantes: A migração inicial do processo de Gestão de Projetos para a Atabaque e a publicação da nova versão foi feita por apenas um participante, que conhecia bem o processo de Gestão de Projetos, mas que não tinha experiência em desenvolvimento Web ou no uso de HTML. A atualização do processo migrado foi feita por três pessoas em diferentes unidades da federação. A autora desta tese não participou deste esforço de migração, apenas da análise de adequação da Atabaque a um novo contexto.

Material de Estudo: Dessa vez, não apenas uma parte de um processo, mas todo o processo de Gestão de Projetos foi migrado.

Resultados e Lições Aprendidas

Mais uma vez a avaliação da solução foi positiva. A solução foi adotada para evoluir todo o processo de gestão de projetos e para criar o processo de Gerenciamento de Portfólio e de Evolução da Maturidade em Gerenciamento de Projetos.

Durante a construção do processo de Evolução da Maturidade e de Gerenciamento de Portfólio, foram utilizadas pessoas distribuídas em três estados, que alteravam simultaneamente o processo. O fato de a solução permitir o controle de versão de cada elemento do processo e a modularização do processo acelerou a sua construção e facilitou o compartilhamento de informações.

A migração do processo de Gestão de Projetos do formato de páginas puramente HTML para o formato estruturado implementado pela Atabaque utilizou cerca de 32 homens/hora. Essa migração envolveu: 4 atividades, 24 sub-atividades, 13 artefatos, 11 papéis, 6 ferramentas, 1 roteiro.

O tratamento de PMPs para o processo de Gestão de Projetos não é feita de forma sistemática, como acontece com o processo de desenvolvimento de software, por isso, não foi possível comparar quantitativamente se houve alteração no percentual de PMPs relacionadas a erros, como foi feito no Estudo de Caso 1. Mas, um indicativo de que Atabaque contribuiu para a qualidade da documentação do processo foram os depoimentos dos participantes do estudo de que o formato da nova documentação do processo facilitava seu entendimento. "O processo de Gestão de Projetos está melhor agora que está documentado no formato do PSDS. É mais fácil navegar pelo processo". "Sugiro a utilização da Atabaque para a documentação de todos os processos da empresa.". "O uso da Atabaque para documentar os processos da empresa pode ser um caminho para aumentar a integração entre eles."

Qualitativamente, a avaliação da Atabaque em relação à eficiência foi positiva: "Durante a homologação do processo as alterações demandadas foram executadas e publicadas durante o evento de forma simples e rápida"; e "A ferramenta é fácil de usar e ficou mais fácil de manter o processo".

A Atabaque está sendo utilizada há um ano para a documentação do processo de Gestão de Projetos.

Ao final do estudo de caso foi constatado que a Atabaque pode ser utilizada para outros processos que não apenas o processo de desenvolvimento de software, como foi o caso do processo de Gestão de Projetos. Está em andamento no SERPRO um projeto para padronizar a plataforma de processos internos. A Atabaque está sendo considerada como alternativa para documentar a descrição dos processos da empresa (a alternativa é a EPF Composer, que passou a ser considerada a partir dos estudos em andamento apresentados nesta tese).

A partir dos estudos de caso conduzidos, foi identificado que a Atabaque pode facilitar a documentação estruturada de processos e sua publicação. A Atabaque pode também, por exemplo, ser utilizada para evoluir não só processos como também modelos de melhoria de processos a exemplo do OMM (Seção 6.3) ou do MR-MPS.

Estudos em andamento

Em paralelo à evolução da solução Atabaque, o grupo de voluntários associados ao projeto Eclipse (ECLIPSE FOUNDATION, 2010) evoluiu a ferramenta EPF Composer, trabalhando algumas das oportunidades de melhoria encontradas em uma versão analisada da ferramenta em 2006.

Uma avaliação preliminar da nova versão da ferramenta indica que esta pode ser mais adequada para a documentação de processos de engenharia de software, em particular devido a

sua quase total aderência ao SPEM. Novos estudos estão sendo conduzidos nesse sentido. Acredita-se que as vantagens percebidas com o uso da Atabaque, em especial a manutenção estruturada e controlada da documentação do processo, repitam-se com o uso da EPF Composer. E que, em complemento, a MPS possa se beneficiar de uma ferramenta mais estável, abrangente e que gera gráficos em formato UML com os fluxos dos processos automaticamente.

Para avaliar a ferramenta EPF Composer, novamente, um ciclo de estudo de caso está sendo conduzido. Em uma primeira iteração, o roteiro para Desenvolvimento Java/Web do PSDS (MALHEIROS, et al. 2010) foi migrado. O roteiro está disponível desde novembro de 2009 e nenhuma PMP relacionada a problemas de exibição do roteiro foi submetida até cinco meses após sua publicação. Ou seja, em princípio o EPF Composer também pode contribuir para a eficácia da documentação. Inicialmente, foram percebidas dificuldades com a gestão de configuração dos elementos do processo. A gestão de configuração do processo gerado com a EPF Composer é mais difícil porque os nomes dos arquivos armazenados são um código numérico utilizado internamente para as ligações entre os componentes do processo. Por outro lado, a possibilidade de associar diagramas do fluxo do processo à sua documentação foi vista como o grande benefício da EPF Composer até o momento.

Um exemplo de fluxos gerados automaticamente com o uso do EPF Composer é apresentado na Figura 44. Neste exemplo, é apresentada uma tela do processo Demoiselle. O processo Demoiselle é um **processo livre para desenvolvimento** de software para o governo com o uso do Framework Demoiselle.

Figura 44: Tela do processo Demoiselle gerada com o EPF Composer

Em paralelo, novas avaliações relacionadas à evolução estruturada e colaborativa de processos poderão ser conduzidas, considerando as evoluções do processo Demoiselle.

6.2.2 Estudos relacionados ao tratamento de PMPs

Nesta seção, são descritos os diferentes estudos experimentais conduzidos a fim de analisar a adequação das estratégias para tratamento de PMPs apresentadas no Capítulo 5.

Quatro estudos experimentais foram conduzidos relacionados ao tratamento de PMPs, todos eles em ambiente industrial. A seqüência dos estudos e uma síntese da evolução dos resultados são ilustradas na Figura 45 e descritos nas seções 6.2.2.1 – 6.2.2.4.

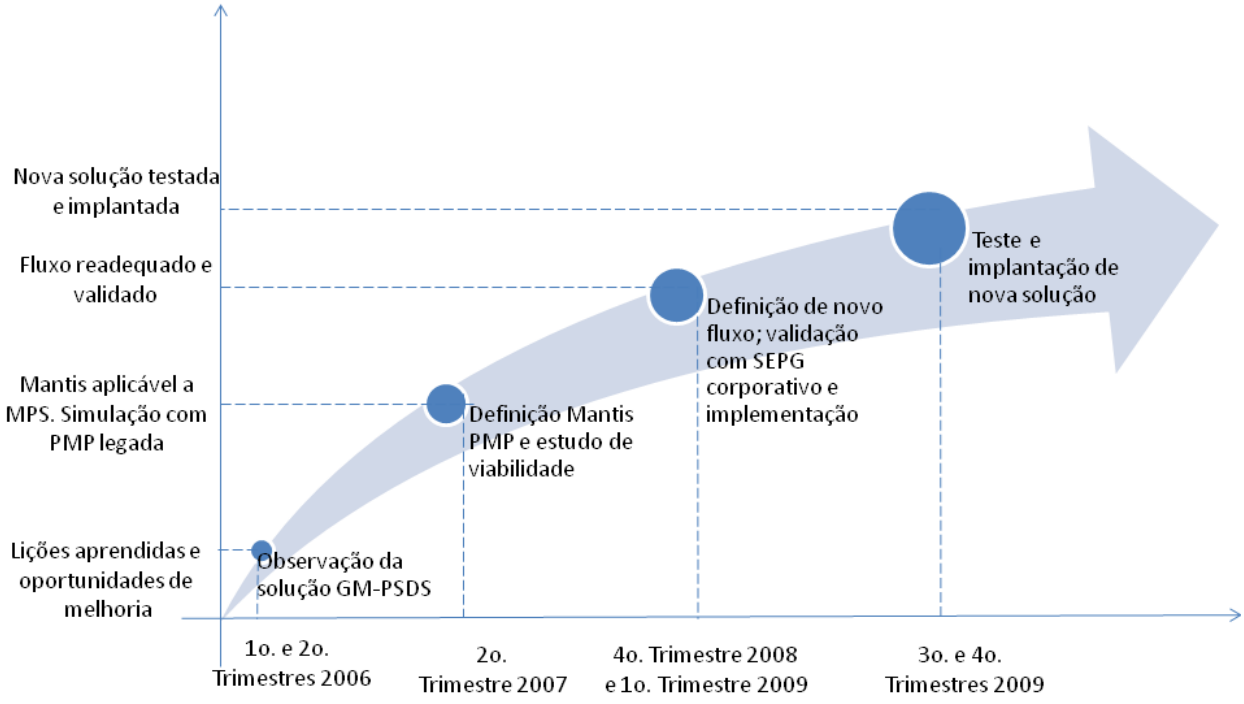


Figura 45: Evolução dos estudos relacionados ao tratamento de PMPs

6.2.2.1 Estudo de caso para analisar a solução GM-PSDS no SERPRO

A solução GM-PSDS foi desenvolvida sobre a plataforma ARS – Remedy (BMC, 2006) para atender à sistemática de Gestão de Mudanças e Suporte ao PSDS definida na disciplina Gestão do Processo da Organização. A GM-PSDS apoiava a automatização das atividades "Gerenciar Propostas de Melhoria de processo" e "Prover Suporte ao Processo", gerenciando todo o ciclo de vida das propostas de melhoria de processos e pedidos de suporte.

A GM-PSDS foi implantada em novembro de 2004. A partir daí, dados da evolução do tratamento de propostas de melhorias do processo ficaram disponíveis em uma base centralizada, permitindo sua exploração. A interface principal da GM-PSDS é mostrada na Figura 46.

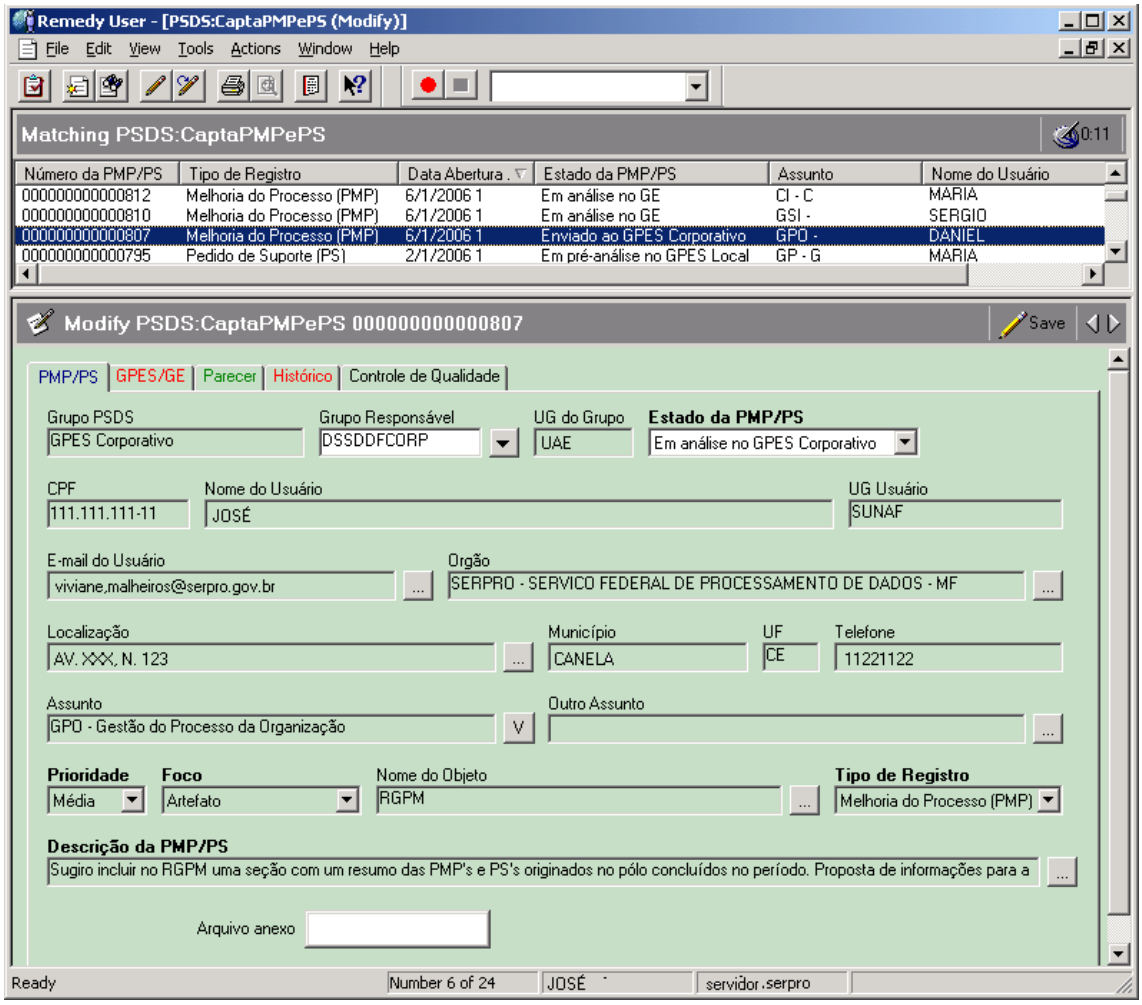


Figura 46: Extrato da tela principal da GM-PSDS (MALHEIROS et al., 2006)

A GM-PSDS podia ser utilizada por usuários de dois perfis:

a) Perfil Corporativo - Esta opção estava disponível apenas aos integrantes dos grupos PSMPS, GPES Locais e GE (Seção 4.2), responsáveis pelo tratamento de PMPs e PSs ao uso de processo, desde o seu cadastro, encaminhamento e registro de pareceres até a sua aprovação ou rejeição, no caso de PMPs, ou solução no caso de PSs.

b) Perfil Usuário - Esta opção estava disponível a todos os usuários do PSDS, acionando-se botão existente na parte superior do *site* do PSDS. Por meio desta facilidade, PMPs e PSs poderiam ser registrados e consultados. Esta modalidade utiliza a interface web do ARS-Remedy e, portanto, não requer instalação de aplicativos ou componentes na estação do usuário.

Por meio da GM-PSDS, todos os dados relativos à MPS são armazenados de forma relacional e podem ser acessados por meio de consultas SQL pré-definidas, ou adaptadas pelo usuário. Ela oferece recursos ao usuário final para encontrar respostas a perguntas como: (a) Qual a situação atual de uma PMP? (b) Para quem a PMP foi enviada? (c) A PMP já foi

analisada? (d) Por quem ela foi analisada? (e) Quando ela foi aberta? (f) Há quanto tempo permanece sem solução? (g) Quantas PMPs foram abertas, aprovadas, por regional e por UG?

Um primeiro estudo de caso foi conduzido no primeiro trimestre de 2006 (Figura 45) e os resultados, reportados em Malheiros et al.(2006), foram insumos para definir a Mantis-PMP (Seção 5.6.3). Esse estudo de caso foi utilizado para monitorar as atividades para tratamento de PMPs e observar a evolução do atendimento das propostas. Lições aprendidas para a melhoria de processos foram registradas.

Objetivo: ***Analisar*** a gestão de propostas de melhoria no SERPRO ***com o propósito de*** caracterizá-la ***quanto à*** eficiência e eficácia ***do ponto de vista*** do usuário do processo, ***no contexto*** da melhoria de processos em uma grande organização. Entende-se por eficiência, para este estudo, o tempo demandado para tratar uma PMP ou um PS (Pedido de Suporte). Por eficácia, entende-se a qualidade da análise e do desdobramento dado às PMP e PS.

Participantes: O estudo de caso foi conduzido por três membros do SEPG corporativo do SERPRO, inclusive a autora desta tese. Todos os três membros possuíam experiência prática na MPS há mais de quatro anos, atuando em diferentes papéis. À época do estudo de caso, atuavam como membros do SEPG há mais de um ano.

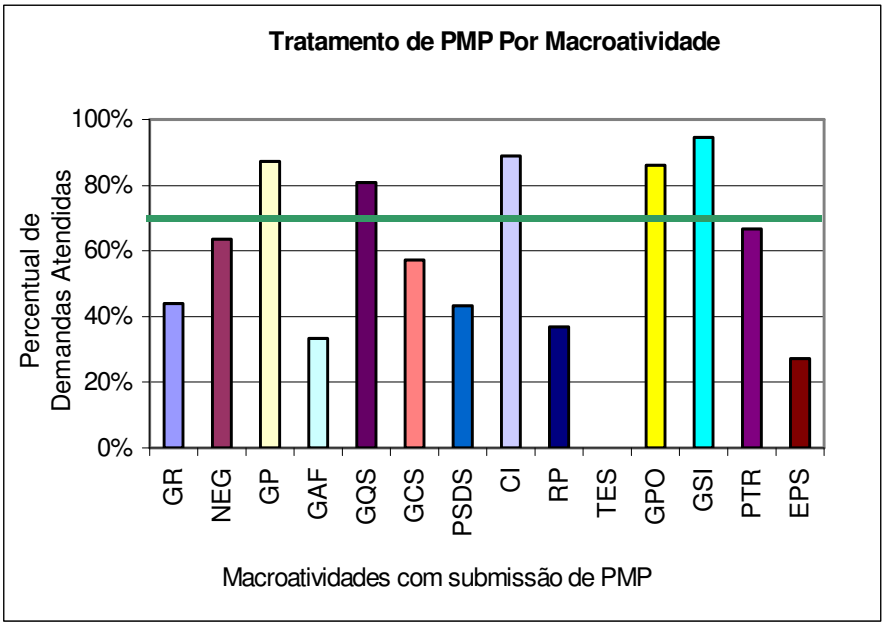
Material para o estudo: Para o estudo de caso, foram selecionados para análise: todas as PMPs, pareceres e PSs registrados na base de novembro de 2004 a janeiro de 2006 e todas as avaliações de atendimento (Controle de Qualidade) solicitadas ou registradas no período. Para avaliar a eficiência do tratamento das propostas duas informações foram consideradas: o tempo corrido entre o cadastro da PMP e sua aprovação e a avaliação qualitativa do usuário relacionada à sua satisfação com o tempo de resposta para o tratamento. Para avaliar a qualidade do tratamento, foi considerada a avaliação dos usuários quanto à satisfação da resposta a PMP. Foi considerada ainda a quantidade de PMPs atendidas por macroatividade para permitir comparação entre as mesmas. Além das bases de dados, a experiência dos três participantes foi considerada. Os três participantes se propuseram a responder as seguintes questões: Quais são os pontos fortes do tratamento de PMPs e PSs? O que pode ser melhorado no tratamento de PMPs e PSs?

Resultados e consolidado das lições aprendidas

Entre novembro de 2004 e janeiro de 2006, foram submetidas 820 solicitações, média de mais de três submissões por dia útil. A quantidade de PMPs e PSs abertos no primeiro ano de utilização da ferramenta GM-PSDS reflete um alto grau de comprometimento dos usuários com a MPS. Desse total, 79% foram PMPs (651) e 21% foram PSs (169). Das 651 PMPs tratadas,

72% foram aprovadas para implantação no PSDS e 28% foram rejeitadas. O alto percentual de aprovação das propostas reflete a pertinência e a qualidade das submissões.

Na Figura 47, o percentual de PMPs atendidas por macroatividades é ilustrado (retrato de janeiro de 2006). A quantidade de PMPs finalizadas (aprovadas ou rejeitadas) correspondia a 63% do total. Esse foi, aproximadamente, o mesmo índice de finalização obtido no último levantamento semestral (agosto 2005). Apesar de esse número ser um reflexo da incidência de PMP de alta complexidade, na época, a meta estabelecida para a próxima medição foi de 70%. Esta meta é mostrada como uma linha horizontal na Figura 47.



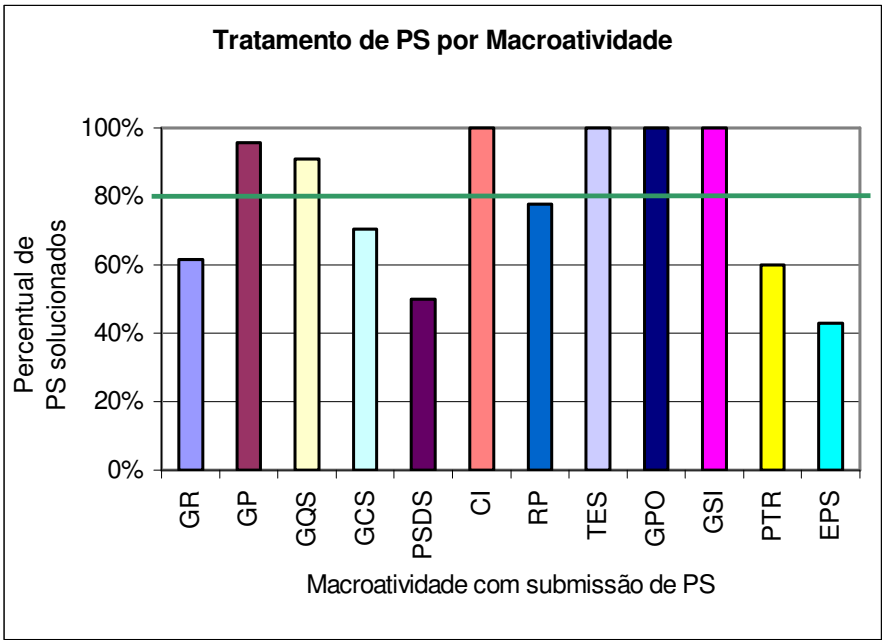
Legenda:
GR - Gestão de Requisitos
NEG – Modelagem de negócio
GP – Gestão de Projetos
GAF – Gestão de Aquisição com Fornecedor
GQS – Garantia da Qualidade de Software
GCS – Gestão de Configuração de Software
PSDS – PMP relacionada com todo o processo
CI – Comunicação Intergrupos
RP – Revisão por Pares
TES – Teste e Homologação
GPO – Gestão do Processo da Organização
GSI – Gestão de Software Integrada
PTR – Programa de Treinamento
EPS – Engenharia do Produto de Software (análise, projeto, implementação, implantação)

Figura 47: Atendimento de PMP por macroatividade (MALHEIROS et al., 2006)

Na Figura 48, o percentual de PSs atendidos por macroatividades é mostrado. O atendimento a PSs foi mais conclusivo, 75% dos PSs foram solucionados rapidamente e apenas 25% estavam aguardando solução. Para o PSDS, o atendimento aos PSs é prioritário, já que a solução rápida de um PS está diretamente relacionada à aderência dos projetos ao processo e pode agregar qualidade ao produto final, ainda no ciclo do projeto. A prioridade dada a esse atendimento no segundo semestre de 2005 se refletiu na melhora do índice de finalização dos PSs, que subiu de 63% para 75%. A meta estabelecida para o próximo ciclo foi de 80% (linha horizontal, na Figura 48).

Os dados das duas figuras foram obtidos durante o mesmo período de tempo. Analisando as duas figuras foi possível perceber que o atendimento a Pedidos de Suporte foi priorizado em todas as macroatividades. Por meio da Figura 47 e da Figura 48 é possível identificar quais grupos estão mais perto ou já atingiram as metas e direcionar ações para os grupos que ainda

estão distantes das metas. Além disso, é possível identificar junto aos grupos com melhor desempenho quais as boas práticas utilizadas e replicá-las para os demais grupos. Os benefícios por tratar essas boas práticas da MPS como lições aprendidas para os próximos ciclos da MPS são discutidos por Santos et al. (2007). Os grupos que mais se destacaram na análise de PMPs geralmente foram os grupos que mais se destacaram no atendimento de pedidos de suporte. A única exceção é o grupo de testes, que teve uma excelente eficácia no tratamento de pedidos de suporte e um fraco desempenho no tratamento de PMPs.



Legenda:
GR - Gestão de Requisitos
GP – Gestão de Projetos
GQS – Garantia da Qualidade de Software
GCS – Gestão de Configuração de Software
PSDS – PMP relacionada com todo o processo
CI – Comunicação Intergrupos
RP – Revisão por Pares
TES – Teste e Homologação
GPO – Gestão do Processo da Organização
GSI – Gestão de Software Integrada
PTR – Programa de Treinamento
EPS – Engenharia do Produto de Software (análise, projeto, implementação, implantação)

Figura 48: Atendimento de PSs por macroatividade (MALHEIROS et al., 2006)

Cerca de 13% do total de propostas foram classificadas como manutenções corretivas, ou seja estavam relacionadas com algum tipo de correção (retrabalho).

A GM-PSDS possuía um recurso para assinalar uma PMP para inclusão na Base de Conhecimentos ou na FAQ (Perguntas Freqüentes) do PSDS. Havia indícios, no entanto, que este recurso não estava sendo utilizado na sua plenitude. Apenas quatro submissões (0,5% do total) foram marcadas para inclusão na Base de Conhecimentos e apenas cinco (0,6% do total) foram marcadas para inclusão na FAQ. Esses indicativos são importantes para a melhoria da Gestão do Conhecimento relacionada ao desenvolvimento de software. Os baixos índices de submissão obtidos apontaram para a necessidade de reforçar a importância desses indicativos para os grupos especialistas.

Como a GM-PSDS refletia a estrutura organizacional de apoio à MPS no SERPRO (Capítulo 4), foi possível identificar a influência da contribuição do atendimento local e do atendimento corporativo no tratamento de propostas. Do total de itens submetidos, 61% foram

analisados localmente, em um primeiro nível de suporte, minimizando ou subsidiando o trabalho de análise do grupo corporativo. Esse número indica um alto nível de conhecimento já disseminado nos grupos de desenvolvimento e demonstra uma grande distribuição do trabalho.

Utilizar a GM-PSDS trouxe ganhos sensíveis para a administração da evolução do PSDS. A partir da sua implantação, toda PMP só era reconhecida pelo PSMPS quando registrada na ferramenta. Assim, nenhuma alteração do processo podia ser processada sem que houvesse PMP associada e aprovada que a justificasse. Essa prática tem o efeito colateral positivo de facilitar o controle da configuração do PSDS.

Existe um desnivelamento em relação aos resultados dos grupos especialistas. Alguns grupos processam as PMPs e PSs mais lentamente que outros e não conseguem envolver rapidamente todos os seus membros. Na maioria das vezes, isso acontece porque os desenvolvedores precisam priorizar as atividades de desenvolvimento de software e não as atividades da MPS. Esse problema pode ser minimizado com alterações no fluxo de PMPs e PSs.

As discussões realizadas na análise das PMPs não ficam associadas às propostas, e sim em e-mail, fazendo com que parte do histórico do tratamento da PMP se perca.

Existem alguns casos de duplicidades de PMPs e PSs, aumentando a quantidade de requisições. Um dos fatores que ocasiona essa duplicidade é o fato dos usuários não terem acesso às propostas já submetidas no momento da submissão de uma nova PMP.

Detectou-se a necessidade de desenvolver funcionalidades para acompanhar a implementação e implantação de uma melhoria no PSDS, após a sua aprovação. Durante o estudo de caso, foi constatado que o acompanhamento até a aprovação (último estado "Aprovada") era insuficiente. A criação dos estados "em implementação", "implementada" e "implantada" poderia ser muito útil.

O atendimento de uma PMP é considerado finalizado quando esta é aprovada ou rejeitada e o de um PS quando este é solucionado. Tanto o **nível de satisfação** com a qualidade quanto com o tempo de atendimento foram medidos por meio de um questionário de avaliação do atendimento (*survey*) aplicado por amostragem aleatória no momento da finalização (aprovação, rejeição ou solução) de uma PMP ou um PS. Para um percentual de cerca de 50% das PMPs e PSs, foi gerada uma mensagem eletrônica para o seu criador questionando sua satisfação em relação: à qualidade da resposta dada (eficácia); e ao tempo decorrido (eficiência) para aprovar/rejeitar a PMP ou responder ao PS. Para ambas as perguntas, eram atribuídas notas que variavam de 1 a 5, crescendo de 1 (insatisfação) até 5 (satisfação plena).

O **nível de satisfação** com o atendimento das PMPs e PSs foi avaliado e considerado muito bom quanto à qualidade da resposta (aproximadamente 4, em média), o que revela que os

grupos estão bem qualificados para atender às demandas. No entanto, identificou-se a necessidade de agilidade nas respostas (avaliação de 3,2, em média). A análise do nível de satisfação com o prazo revelou que seria necessário analisar o tempo médio necessário para solucionar as propostas de melhoria e pedidos de suporte e onde estavam os gargalos (Seção 4.2.3, Figura 28). A tempestividade no tratamento das PMPs é fator importante para aumentar a satisfação dos desenvolvedores e a credibilidade da MPS.

Em relação especificamente à ferramenta, a GM-PSDS foi construída sob uma plataforma proprietária e demanda instalação de aplicativo no lado cliente para sua utilização, dificultando a sua distribuição e gerenciamento para os participantes da MPS distribuídos. Além disso, a solução funciona apenas no sistema operacional Windows, o que dificulta o seu uso em um cenário de expansão do uso do Linux como sistema operacional.

6.2.2.2 Definição da solução Mantis-PMP e estudo de viabilidade

A partir das lições aprendidas com observações da GM-PSDS, foram definidas novas estratégias para tratar propostas de melhoria de processo. Para estudar a viabilidade das novas estratégias de tratamento de melhoria, o novo fluxo de tratamento e as novas atividades foram adaptados no Mantis, dando origem à solução Mantis-PMP (a versão atual da solução Mantis-PMP é apresentada na Seção 5.6.3).

A concepção, desenvolvimento e implementação da solução Mantis-PMP e o estudo da sua viabilidade envolveram os seguintes passos (MALHEIROS et al., 2007b):

1. Observação de pontos fortes e oportunidades de melhoria da ferramenta proprietária GM-PSDS e do processo de melhoria adjacente (Seção 6.2.2.1);
2. Observação de modelos para evolução de software, de acordo com a concepção de desenvolvimento de software livre;
3. Generalização da experiência em MPS do SERPRO para que a solução pudesse ser adotada em outras empresas ou pela comunidade de software livre. Para tanto, foram comparadas as oportunidades de melhoria encontradas no SERPRO com os fatores críticos de sucesso identificados na revisão sistemática na literatura (Capítulo 3);
4. Definição do fluxo para Propostas de Melhoria de processo e concepção da solução (Seção 5.6.3);
5. Especificação dos requisitos e da arquitetura para apoiar a solução;
6. Implementação da solução, a partir de uma adaptação do Mantis; e
7. Teste e implementação de pequenas melhorias na solução.

Objetivo: ***Analisar*** o uso de Mantis-PMP para gerenciar propostas de melhoria de processo ***com o propósito de*** caracterizá-lo ***quanto a*** sua eficiência e eficácia, ***do ponto de vista*** do membro do SEPG, ***no contexto*** de uma melhoria distribuída de processos em uma organização de grande porte.

Hipótese: A utilização de uma adaptação do Mantis-PMP pode agilizar (reduzir o tempo) o tratamento de PMPs, eliminando passos desnecessários e oferecendo uma interface que já seja conhecida dos desenvolvedores. Em relação à eficácia, será observada a viabilidade de acompanhar PMPs após a aprovação, parte do fluxo não implementada na solução anterior, com o uso da Mantis-PMP.

Participantes: Três pessoas, com diferentes habilidades, participaram dessa adaptação. Os três participantes são colaboradores do SERPRO; sendo, na época, um membro do SEPG corporativo (a autora desta tese), um gerente sênior, e um desenvolvedor. O desenvolvedor possuía experiência em adaptações do Mantis, tendo feito uma monografia de pós-graduação sobre a seleção de ferramentas de tratamento de erros (CUNHA, 2005). O gerente sênior, além de contribuir para definir a Mantis PMP, implementou a ferramenta Atabaque (Seção 5.6.2).

Material do Estudo: Novo fluxo de tratamento de PMPs, implementado como uma adaptação do Mantis (Mantis-PMP) e propostas de melhoria reais que foram transcritas em simulações de uso da Mantis-PMP.

Resultados e Consolidado das lições aprendidas

A adaptação proposta foi a terceira forma de adaptação do Mantis à realidade do SERPRO (Figura 49).

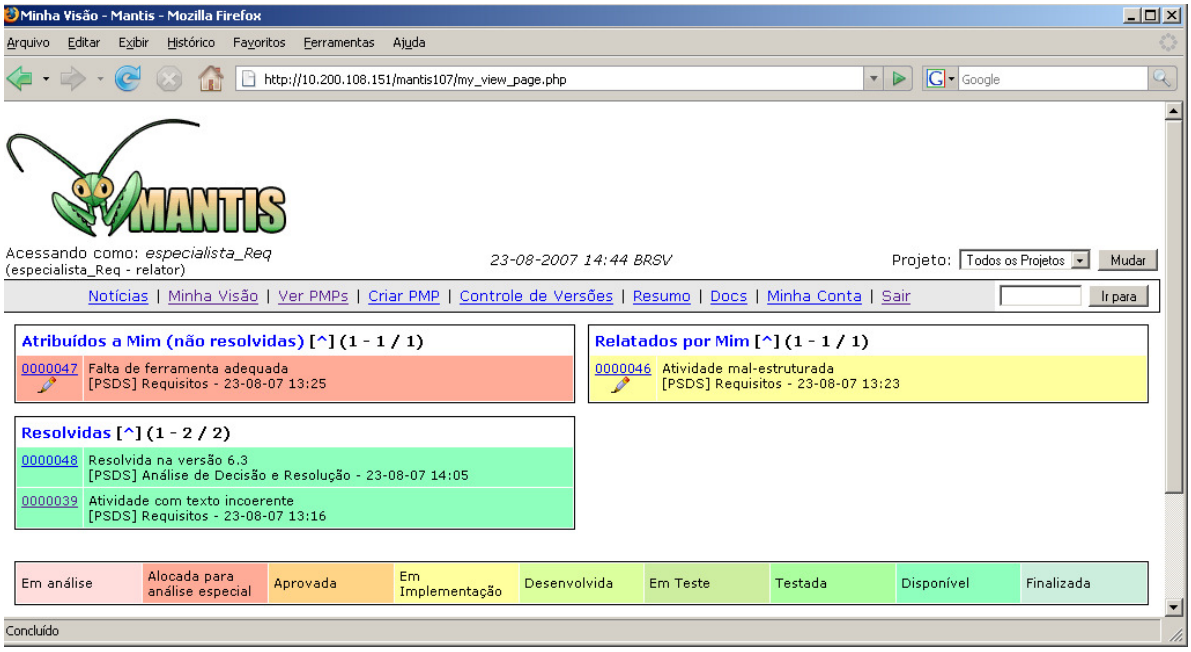


Figura 49: Tela "Minha Visão" de Mantis-PMP (MALHEIROS et al. 2007b)

O Mantis já era utilizado na empresa para tratar erros na fase de testes e para apoiar a etapa de homologação de sistemas. Além das facilidades no desenvolvimento da nova solução, a seleção da nova arquitetura traria mais vantagens, a saber:

1. Portabilidade do código produzido, além de permitir a sua integração com diversos ambientes operacionais;
2. Implementação utilizando uma ferramenta de desenvolvimento gratuita e que atende a um padrão não proprietário;
3. Semelhança da interface do Mantis-PMP com o Mantis, de uso difundido entre desenvolvedores da comunidade de software livre e do SERPRO;
4. Agilidade na implementação e evolução, em função do reúso; e
5. Minimização de gastos adicionais com treinamentos, possibilitando o uso quase imediato da solução proposta. Apenas uma palestra para apresentação das funcionalidades disponíveis seria suficiente para capacitar os usuários.

As principais atividades previstas para o tratamento de PMPs nessa primeira versão da Mantis-PMP estão representadas na Figura 50. Por meio desta, é possível identificar, além do fluxo de atividades, o papel responsável pela execução de cada uma dessas atividades.

O diagrama de estados proposto para essa nova solução para tratamento de PMPs é apresentado na Figura 51. Em relação ao fluxo implementado na GM-PSDS, este fluxo traz algumas novidades baseadas nas lições aprendidas com o uso da GM-PSDS, entre elas:

- a possibilidade de acompanhar PMPs mesmo após a sua aprovação, com a criação dos estados “Em implementação”, “Disponível” e “Finalizada”;
- a eliminação dos estados “Enviado para o GPES corporativo” e “Enviado para o grupo especialista”; e
- a possibilidade dos usuários visualizarem as PMPs e consultá-las antes de gerarem uma nova PMP.

Para analisar a viabilidade de utilização do novo fluxo, os três autores da solução simularam o seu uso, assumindo diferentes papéis relacionados à MPS (membro de grupo especialista, membro do SEPG, etc.) e cadastrando PMPs reais, extraídas da ferramenta GM-PSDS. Para simular, alguns passos foram estabelecidos, por exemplo: usar a ferramenta em diferentes navegadores (Internet Explorer e Firefox); verificar a existência de uma PMP antes de cadastrá-la; consultar todas as PMPs implementadas para uma versão, etc.

O objetivo desse estudo foi verificar a viabilidade de utilizar a adaptação do Mantis para implementar a Mantis-PMP e conhecer a viabilidade de utilizar novos fluxos para tratar PMPs.

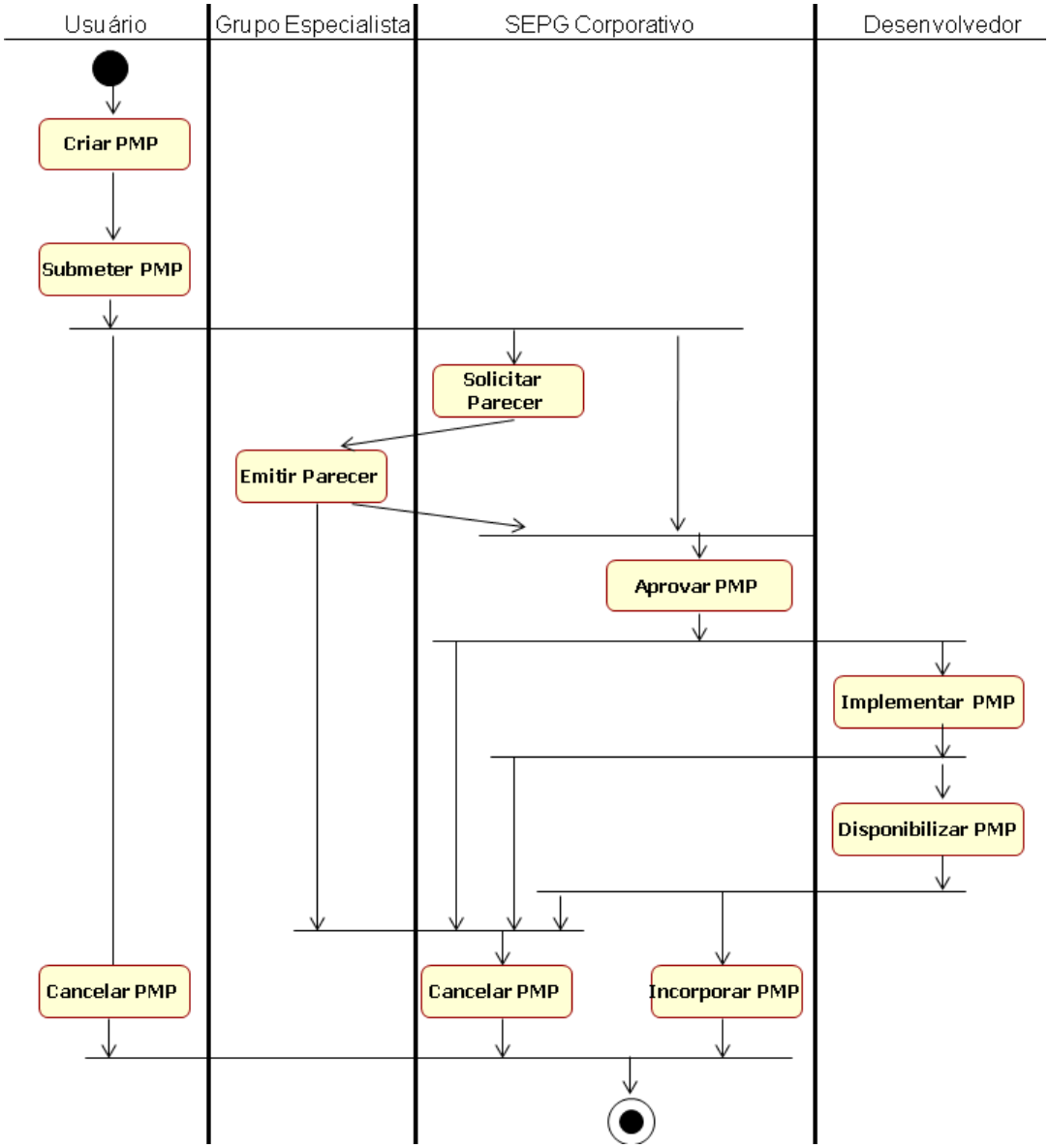


Figura 50: Diagrama das principais atividades para tratamento de PMPs

O Mantis mostrou-se viável para implementar o tratamento de PMPs e a solução pôde ser utilizada tanto por meio do Internet Explorer quanto por meio do Firefox. Foi possível utilizar os sistemas operacionais Windows e Linux para executá-la. É possível configurar a Mantis para tratar PMPs e, ainda assim, manter a interface e o fluxo bem parecidos com o tratamento de erros, minimizando os impactos de adotar novas ferramentas. A possibilidade de gerar relatórios diretamente na Mantis-PMP é uma facilidade para gerir o tratamento de PMPs.

Existe equivalência entre o fluxo possível de ser implementado no Mantis e as atividades previstas no processo de Gestão de Processos.

Ao simular o cadastro e tratamento de algumas PMPs reais na Mantis-PMP, percebeu-se que para algumas PMPs de rápida implementação não havia a necessidade de passar pelos

estados “Aprovada” e “Em implementação”, podendo uma PMP ir diretamente do estado “Em análise” para o estado “Disponível”.

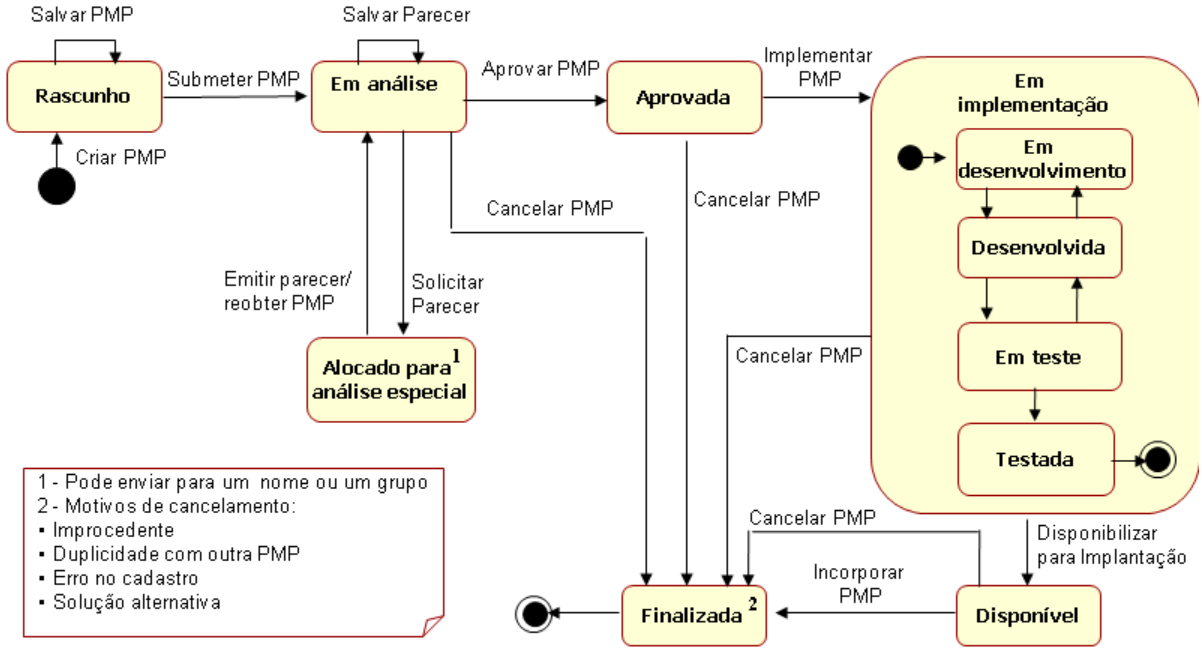


Figura 51: Diagrama de estado para tratamento de PMP - 1ª. versão da Mantis-PMP

O detalhamento de todas as fases das PMPs “Em implementação” pode complicar o fluxo de tratamento de PMPs. O uso do estado “Em implementação” pode ser suficiente para acompanhar o progresso de uma PMP.

Não há a possibilidade de um usuário, ao consultar uma PMP, acrescentar sua opinião sobre a proposta, sem que ele tenha sido demandado por um parecer. Ao encontrar uma PMP semelhante a uma PMP que seria cadastrada, pode ser necessário acrescentar comentários à mesma, complementando-a.

Foi possível consultar PMPs já cadastradas antes de cadastrar uma nova PMP.

Ao simular o uso da Mantis-PMP, sentiu-se necessidade de integrar funcionalidades *Wiki* para discutir as PMPs, pois dessa forma, a evolução de processos poderia beneficiar-se mais ainda da forma livre e colaborativa de trabalho adotada pelas comunidades de software livre.

6.2.2.3 Redefinição do fluxo de PMP e validação com SEPG corporativo

Após o estudo de viabilidade que confirmou a possibilidade de utilizar a Mantis-PMP para o tratamento de PMPs e a pertinência de acompanhar o progresso da PMP pós-aprovação (“Em implementação”, “Disponível” e “Finalizada”), a proposta foi submetida a todo SEPG corporativo para um estudo mais aprofundado da adequação e efetividade da nova solução.

Objetivo: ***Analisar*** a Mantis-PMP e seu novo fluxo para gerenciar propostas de melhoria de processo ***com o propósito de*** caracterizá-lo ***quanto a*** adequação, ***do ponto de vista*** do SEPG corporativo, ***no contexto*** de uma melhoria distribuída de processos em uma organização de grande porte.

Hipótese: O novo fluxo proposto por Mantis-PMP substitui com vantagens o fluxo da GM-PSDS e está adequado às necessidades da MPS de uma organização de grande porte com mais de 3.000 desenvolvedores distribuídos.

Participantes: Nove pessoas, todo o SEPG corporativo do SERPRO à época, participaram desse estudo, incluindo a autora desta tese.

Material do Estudo: Lições aprendidas no estudo de viabilidade da Mantis-PMP e fluxo redefinido a partir da experiência com o tratamento de PMPs adquirida pelo SEPG corporativo.

Resultados e Lições Aprendidas

A partir do fluxo implementado na adaptação da Mantis-PMP e das lições aprendidas no estudo de viabilidade, foi desenhado um novo fluxo do tratamento da melhoria. Esse novo fluxo foi apresentado na Figura 38, Seção 5.6. O novo fluxo foi discutido de forma colaborativa e distribuída pelo SEPG corporativo. Um ambiente *Wiki* foi utilizado para as discussões dos membros do SEPG corporativo, que estava espalhado entre as cidades: Rio de Janeiro, São Paulo, Brasília e Ribeirão Preto. Na Tabela 11, o mapeamento entre os três fluxos utilizados nos estudos é apresentado.

Tabela 11: Comparação entre os estados de cada fluxo para tratar PMPs

Fluxo 1 (GM-PSDS)	Fluxo 2 (estudo de viabilidade Mantis-PMP)	Fluxo 3 (validado pelo SEPG corporativo e atual)
Em Registro	Rascunho	Em registro
Em pré-análise	-	Em pré-análise
Enviado ao GPES corporativo	-	-
Em análise no GPES corporativo	Em análise	Em análise
Enviado ao grupo especialista	-	-
Em análise no grupo especialista	Alocado para análise especial	-
Aprovada	Aprovada	Aprovada
-	Em implementação + Desenvolvida, Em teste, Testada	Em implementação
-	Disponível	Implementada/ Aprovada para *release*
Rejeitada	Finalizada	Rejeitada/Cancelada/ Implantada

Em relação ao fluxo proposto no estudo de viabilidade anterior, esse fluxo traz a possibilidade dos usuários registrarem colaborações a uma PMP em diferentes momentos do fluxo da PMP. Os subestados do estado "Em implementação" foram suprimidos e são opcionalmente informados em um campo de observação. O estado "Disponível" foi dividido em

dois estados "Implementada" e "Aprovada para *release*". O grupo avaliou que não seria necessário um acompanhamento tão detalhado dos estados intermediários da PMP. A aprovação para *release* indica que a alteração implementada foi homologada e está pronta para entrar em produção. Para completa adequação à necessidade de distribuição da decisão relacionada à MPS e inclusão dos interesses corporativos e locais, o estado "Em pré-análise" voltou a ser incorporado ao fluxo.

Foi criada a possibilidade de converter uma PMP do estado "Em análise" para o estado "Implementada". Essa alternativa no fluxo de tratamento de PMPs pode ser muito útil para as propostas que são muito rapidamente implementadas e não exigem testes elaborados.

O grupo achou mais adequada a separação de alguns estados finais da PMP. Assim, em substituição a um único estado final "Finalizada", foram criados os estados: "Cancelada" e "Implantada". Além disso, foi criada a possibilidade de reverter o estado final "Rejeitada" e aberta a possibilidade de solicitar revisão da rejeição.

O Fluxo 3 (Figura 38, Seção 5.6.3) é o praticado atualmente por meio de uma adaptação do Mantis no SERPRO, dados os benefícios identificados.

6.2.3 Estudos relacionados ao uso de um Ambiente de Desenvolvimento Colaborativo (ADC) para a MPS

Dois estudos foram utilizados para analisar o uso de mecanismos de comunicação, por meio do uso de um ADC para a MPS. Esses estudos são ainda estudos de viabilidade para identificar as possibilidades de contribuição de um ADC para a MPS e ainda não foram formatados como os demais estudos.

O primeiro foi um estudo de viabilidade, quando foi criado um projeto em um ADC adaptado às necessidades da MPS. Esse estudo foi conduzido em duas etapas, primeiro com a simulação de um projeto da MPS em um servidor do Gforge[53] criado para viabilizar a geração de um protótipo. A tela principal desse protótipo foi apresentada na Figura 39. Na segunda etapa, os mecanismos de comunicação foram criados em um projeto para a MPS em um ADC utilizado internamente no SERPRO. O ADC utilizado no SERPRO é uma adaptação do Gforge, o Colab.Serpro. O Colab.Serpro integra funcionalidades como fóruns de discussão, administração de projetos, acompanhamento de atividades, monitoramento de erros, Wiki, publicação de informações, lista de perguntas freqüentes, entre outras. Durante o estudo de viabilidade algumas

[53] GForge é um software para desenvolvimento colaborativo para a comunidade de software livre, baseado na versão 2.1 do SourceForge. Fornece um sistema de desenvolvimento com sítio de projetos e ferramentas de comunicação entre membros do time de desenvolvimento.

dessas funcionalidades foram configuradas para a MPS. As funcionalidades de monitoração de erros e requisição de novas funcionalidades foram desabilitadas, já que ambas as necessidades foram tratadas por Mantis-PMP (Seção 6.2.2). Um *link* para a Mantis-PMP foi acrescentado ao projeto de melhoria para garantir o atendimento ao requisito de ter acesso à informação acerca da MPS por meio de um ponto de acesso único. As funcionalidades de fóruns de discussão, listas de distribuição de mensagens de correio eletrônico, últimas notícias, vagas para projetos, enquetes, Wiki e gestão de documentos foram configuradas de acordo com as características da MPS.

Foram definidas listas de distribuição relacionadas à estrutura organizacional da MPS: lista para o SEPG corporativo; lista para o grupo especialista (uma lista para cada disciplina).

Quanto à gestão de documentos, foram criadas as pastas: "gestão da MPS", que está dividida entre planejamento e acompanhamento e provê acesso, por exemplo, para o plano da MPS; "referências", que aponta para os *sites* e arquivos das referências utilizadas para construir o processo; "apresentações", que inclui as apresentações geradas sobre o processo; "processo", que inclui acesso a última versão publicada do processo; e "orientações", que oferece acesso a arquivos e *links* com orientações sobre como utilizar o processo, por exemplo, como baixar o EFP Composer e configurá-lo para editar o processo. O principal benefício dessa funcionalidade foi manter disponível informações relacionadas à MPS, como o plano de projetos da MPS.

A funcionalidade últimas notícias não se mostrou adequada para a MPS durante o estudo de viabilidade. Poucas pessoas acessam sistemática e freqüentemente o Colab.Serpro e quando o fazem, na maioria das vezes, acessam seus projetos diretamente. Os canais de comunicação gerais da empresa parece garantirem um maior alcance dos desenvolvedores. Vale ressaltar que o ambiente Colab.Serpro é novo e ele mesmo ainda passa por um período de internalização. Esse fato pode ter impactado negativamente o uso da funcionalidade últimas notícias.

A funcionalidade vagas para projetos também se mostrou um pouco limitada. Foi identificado apenas um acesso espontâneo a essa funcionalidade. As consultas às vagas por projeto, via de regra, foram precedidas de algum outro tipo de divulgação.

Aparentemente os fóruns podem ajudar as discussões temáticas da MPS. Exemplo disso foi a proposição feita por um desenvolvedor para incluir um tema para discussão: "Segurança no Desenvolvimento" no fórum.

O segundo estudo foi iniciado recentemente. Foi criado um ambiente para a MPS do processo Demoiselle no Sourceforge, por meio da configuração de um projeto específico para melhorar o processo (Figura 52).

Esse ambiente provê um acesso único ao processo; às informações sobre a sua melhoria e a mecanismos que facilitam a comunicação e a colaboração entre os interessados, entre eles: o

repositório do processo, onde os grupos especialistas e o SEPG corporativo podem encontrar ou disponibilizar uma nova versão do "código-fonte" do processo em formato XMI[54] (Figura 53); arquivos relacionados à MPS (Figura 54), a exemplo da política de publicação do processo; fórum, a partir de uma integração do Sourceforge com a ferramenta phpBB[55] listas de discussão ("demoiselle-proc-users" e "demoiselle-proc-sepg").

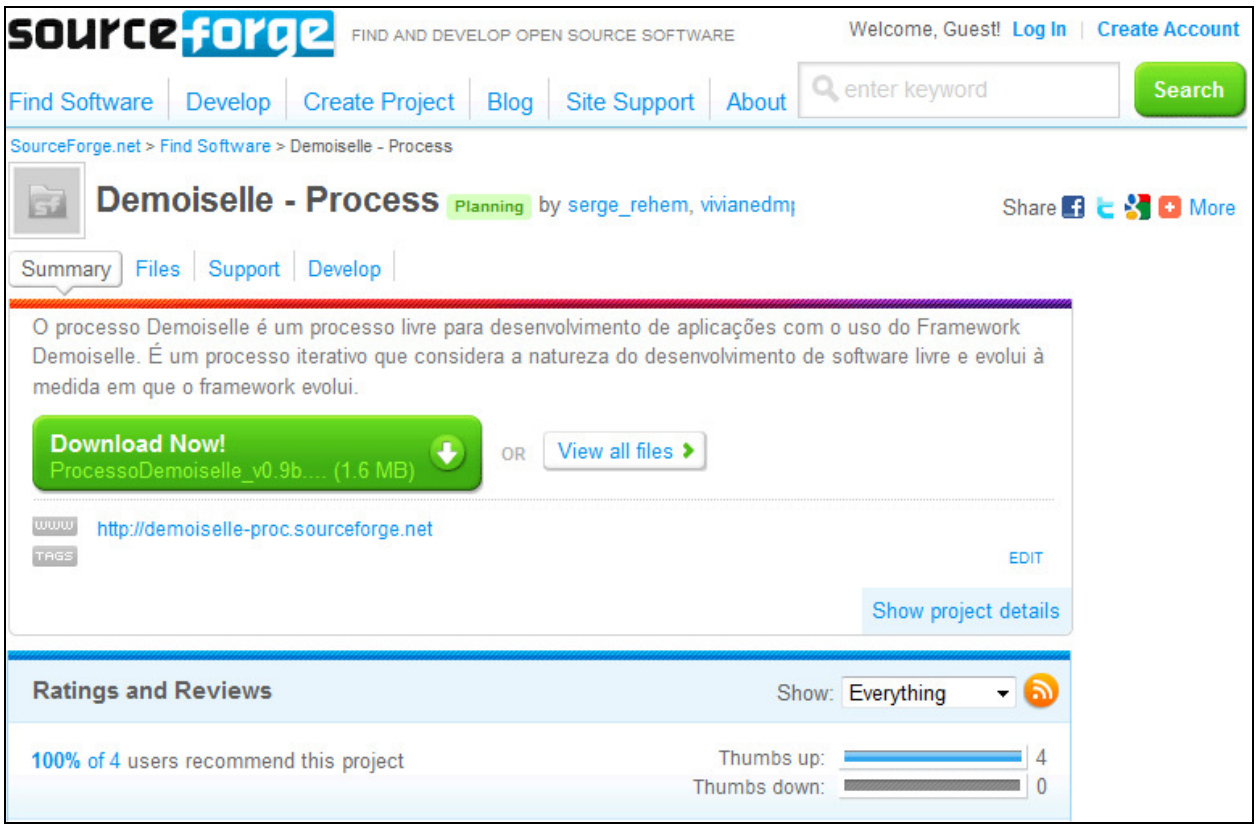


Figura 52: Tela do projeto "processo Demoiselle" no Sourceforge

File	Rev.	Age	Author	Last log entry
ProcessoDemoiselllePlugin/	69	2 days	Maria	Processe Demoiselle Plugin 0.9b
configurations/	40	11 days	João	Processe Demoiselle Plugin
.project	19	2 weeks	Maria	Processe Demoiselle Plugin
demoiselle-logo.jpg	29	13 days	João	Processe Demoiselle Plugin
library.xmi	22	2 weeks	Maria	Processe Demoiselle Plugin
sobre.htm	67	3 days	João	Processe Demoiselle Plugin

Figura 53: Detalhes do repositório do SVN do processo Demoiselle

[54] XMI - *XML Metadata Interchange* (http://www.omg.org/technology/documents/formal/xmi.htm)
[55] www.phpbb.com.br

Figura 54: Detalhe da página de acesso aos arquivos da MPS

As perspectivas de uso do ambiente da MPS do processo Demoiselle são animadoras. Desde que a atividade do projeto começou a ser monitorada por meio de mecanismos para esse fim, integrados ao ADC, 211 visitantes únicos acessaram o projeto, com uma média de 12 visitantes ao dia no período observado.

6.2.4 Estudos relacionados ao uso de pilotos para MPS

Para estudar e incorporar métodos ágeis em um processo tradicional, estratégias de MPS colaborativas e distribuídas foram aplicadas. Uma das metas estabelecidas no direcionamento organizacional do SERPRO foi melhorar o processo de software com foco em qualidade e produtividade e, em particular, tornar o processo mais ágil. Alinhado a esse direcionamento, o programa de MPS estruturou um projeto específico para desenvolver e avaliar melhorias que alterassem a forma de trabalho incorporando, quando adequado, práticas dos métodos ágeis ao processo já instituído. As experiências em MPS, adquiridas com a execução desse projeto, que inclui a condução de projetos piloto, são reportadas nesta seção. Não está no escopo desta tese apresentar as lições aprendidas sobre a incorporação de métodos ágeis, que foram reunidas em Malheiros et al. (2010b). Apenas as lições aprendidas relacionadas à MPS são apresentadas aqui.

Segundo Lindvall et al. (2004), o desafio de integrar efetivamente práticas ágeis a um ambiente organizacional é diferente de apenas aplicar práticas ágeis a um projeto especifico. Incorporar novas práticas em um processo e em um sistema de qualidade existentes, independentemente dos benefícios da nova metodologia, requer adaptações. Para estudar e incorporar práticas ágeis ao PSDS, toda a estrutura organizacional de MPS (Capítulo 3) foi utilizada. Essa estrutura de MPS foi útil para incluir contribuições dos desenvolvedores no processo, pois a mesma é distribuída e permeia toda a organização. Os membros dos grupos especialistas, também distribuídos, trabalharam de maneira conjunta para discutir a melhoria. Para validar o processo melhorado, projetos piloto foram conduzidos e seus resultados foram utilizados para as novas versões do processo.

Um arcabouço para trabalhar com roteiros como mecanismo para adaptação do processo às necessidades dos usuários foi discutido e a idéia de incorporar as metodologias ágeis foi

desenhada. Um roteiro é uma organização de elementos do PSDS (atividades, papéis, orientações, etc.) em um fluxo adequado para uma necessidade específica. O arcabouço para roteiros e as metodologias ágeis foram discutidos com o SEPG corporativo inicialmente e, na seqüência, com representantes dos grupos especialistas. Como alguns desenvolvedores na empresa detinham conhecimento sobre metodologias ágeis, eles também foram convidados a opinar. Para isso, a proposta foi colocada em um ambiente *Wiki*. O ambiente *Wiki* ajudou nas discussões iniciais e permitiu o envolvimento de mais colaboradores.

Quatro roteiros foram identificados inicialmente: (i) iterativo, um roteiro para desenvolvimento Java/Web com apoio de um *framework*; (ii) pequenas manutenções; (iii) desenvolvimento de *data warehouse*; (iv) desenvolvimento de componentes. Para outros projetos não há roteiros específicos e o fluxo principal do PSDS é utilizado como referência. O roteiro Java/Web foi escolhido como sendo o primeiro roteiro a ser desenvolvido.

Toda a melhoria relacionada à incorporação de métodos ágeis foi conduzida de forma iterativa como proposto na Seção 5.3. A maior parte dos trabalhos foi conduzida à distância e por meio de mecanismos de colaboração. Mas, por se tratar de uma discussão de paradigmas, percebeu-se que apenas os tais mecanismos não seriam suficientes. Duas reuniões presenciais foram agendadas para refinar as discussões. Uma síntese da evolução dessa melhoria é apresentada na Figura 55.

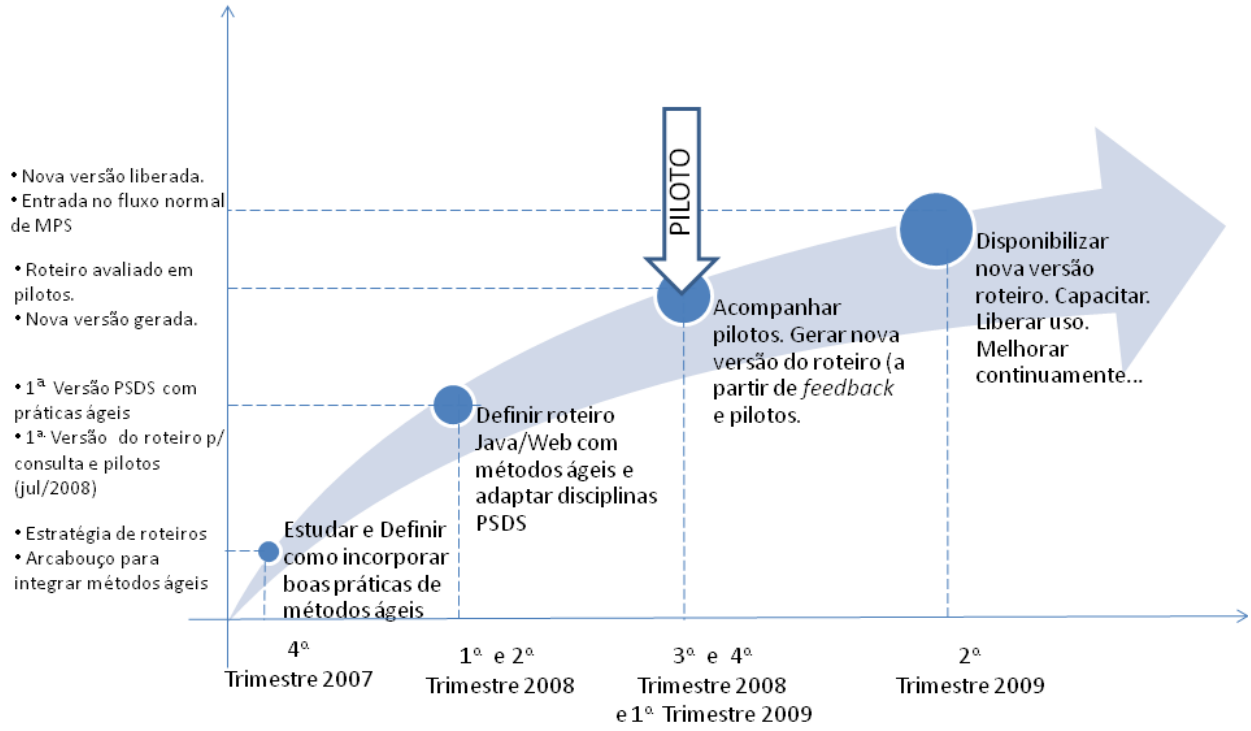


Figura 55: Exemplo de MPS com estudo piloto, incorporação de métodos ágeis

A seqüência de eventos para incorporar os roteiros e os métodos ágeis incorpora pilotos com equipes de desenvolvedores para avaliar as mudanças para o processo, antes de introduzi-las de fato no processo em produção. O uso de pilotos está alinhado com as recomendações de Gremba e Myers (1997) e de Ambler (2005).

A estratégia de utilizar projetos piloto associados ao passo "3 - Desenvolver a iteração de MPS" (Seção 5.4) é recomendada quando uma intervenção no processo se propuser a alterar significativamente a forma de trabalho.

Um resumo do primeiro piloto conduzido é apresentado nesta seção.

Objetivo: Em relação a MPS, o objetivo foi ***analisar*** o uso de projetos-piloto para a MPS ***com o propósito*** de avaliar ***com respeito à*** eficácia ***do ponto de vista*** do SEPG ***no contexto de*** uma alteração significativa no processo.

Hipótese (H1): A utilização de projetos piloto para avaliar alterações significativas em processos pode melhorar os resultados da MPS.

Participantes: Equipe de desenvolvimento de software composta por membros da indústria, executando os seus papéis comuns no desenvolvimento de software. A equipe foi composta de oito pessoas diretamente envolvidas no projeto. Um membro do SEPG corporativo foi designado como consultor do roteiro e acompanhou a evolução do projeto.

Material de estudo: Primeira versão do roteiro Java/Web, publicada e integrada no PSDS, para uso da equipe de desenvolvimento do projeto piloto e consulta das demais equipes de desenvolvimento.

Resultados e Lições aprendidas relacionadas à MPS

O primeiro projeto-piloto foi o desenvolvimento de um software em Java para geração e disponibilização de informações fiscais. A duração total do projeto foi de dezesseis semanas e seis iterações foram programadas.O ciclo de vida foi iterativo e seguindo os princípios ágeis de entregas rápidas e curtas. Do total de oito pessoas na equipe de desenvolvimento, no mínimo seis delas participaram ativamente de cada iteração. Os desenvolvedores não possuíam experiência preliminar em métodos ágeis e atividades freqüentes de *mentoring* foram previstas. O mentor (representante do SEPG corporativo) conversou com a equipe semanalmente e encontraram-se presencialmente três vezes durante os quatro meses. A cada encontro, lições aprendidas para melhorar o processo eram coletadas. Para encontros presenciais foram gerados relatórios, nos quais as seguintes informações eram coletadas: Panorama geral; Observações por macroatividades; Problemas identificados no uso do processo; e Lições aprendidas.

O uso de piloto deu mais segurança à equipe de MPS para prosseguir com a intervenção no processo e viabilizou experimentar e avaliar uma nova solução em um contexto real, sem que

a organização como um todo fosse impactada. O uso de piloto permitiu eliminar alguns problemas identificados na primeira versão proposta, antes que a solução fosse replicada para o desenvolvimento. Após o primeiro piloto, uma segunda versão do roteiro foi gerada e validada em um segundo piloto. Apenas após a validação da segunda versão, o roteiro foi liberado para uso pelas unidades.

Durante todo o tempo dos projetos-piloto, o novo roteiro ágil esteve disponível apenas para consulta pelas outras equipes de desenvolvimento. Apesar de não poder ser utilizado para desenvolvimento de soluções para os clientes, que não as soluções dos pilotos, o roteiro sendo avaliado ficou sempre disponível para consulta por todos os desenvolvedores. O objetivo de manter o roteiro disponível para consulta foi dar maior visibilidade às mudanças e facilitar o entendimento das mudanças em curso.

6.3 Utilização da estratégia para a evolução do OMM

Nesta seção, é descrita a estratégia de evolução do OMM, uma adaptação da estratégia ColabSPI. Aqui é descrito como as comunidades interessadas no OMM administram, influenciam e colaboram com a evolução do modelo.

O OMM é um dos principais resultados da Atividade 6 do projeto Qualipso (Seção 2.5.3). Além do modelo propriamente dito, outros resultados importantes são: (i) o conjunto de ferramentas recomendadas para apoiar o uso do modelo; (ii) avaliações de projetos de Open Source baseadas no OMM; e (iii) a estratégia e a plataforma para evoluí-lo. O OMM, as ferramentas de apoio e o processo de avaliação são genericamente nomeados "OMM *process suite*" (WITTMANN et al., 2009). Para evoluir a OMM *process suite*, foi necessário definir uma estratégia (princípios, comunidades e passos) e uma plataforma para viabilizar a evolução colaborativa e distribuída do modelo. Por se tratar de um modelo de maturidade para desenvolvimento de software livre, que deve ser evoluído de maneira distribuída e colaborativa pela comunidade, foi identificada a oportunidade de adequar a estratégia de MPS proposta para o próprio OMM.

Mais uma vez os elementos que compõem a estratégia são: **princípios**, **passos**, **infraestrutura** e **estrutura organizacional**, agora das comunidades interessadas no OMM. A adaptação da ColabSPI para evoluir o OMM está descrita em Malheiros e Maldonado (2009b), disponível no portal do Qualipso. Os princípios que norteiam a evolução do modelo são:

- a OMM *process suite* é livre e todos participam de acordo com as mesmas regras;

- A evolução do modelo acontecerá de forma iterativa. A evolução do modelo será norteada por um plano (*roadmap*);
- Os usuários do OMM participarão da sua evolução. A participação na evolução é baseada no mérito; e
- A comunicação aberta e transparente será priorizada. As informações a respeito da evolução do modelo (planos, discussões, artefatos de gestão, etc.) são públicas e facilmente acessíveis. Canais de comunicação serão disponibilizados.

6.3.1 Organização das comunidades do OMM

Da mesma forma que para a ColabSPI, foram mapeados os principais grupos de interesse no modelo OMM (Figura 56). Como a evolução do OMM não está inserida em uma empresa, a cadeia administrativa é representada por um único grupo responsável pelas decisões da evolução. O equivalente ao SEPG corporativo da ColabSPI é o OPG (*OMM Process Group*).

O **OPG** é o grupo direcionador do modelo, responsável por administrar sua evolução, incluindo: a coordenação da participação dos outros membros; a avaliação de mudanças significativas no modelo; e a definição da visão geral do OMM. Esse grupo é responsável pelo *roadmap* do modelo, responsabilizando-se pelo planejamento das atividades anuais e pelo acompanhamento da execução dessas atividades. Inicialmente, esse grupo será formado por membros dos centros de competência do Qualipso (cada centro indica um membro). A formação do grupo poderá variar de acordo com a contribuição contínua de voluntários. As regras de transparência, meritocracia e participação aberta perseguidas na evolução de projetos de software livre são válidas para a evolução do OMM.

Figura 56: Comunidades para evoluir o OMM (MALHEIROS e MALDONADO, 2009b)

Enquanto o OPG equivale aos "administradores" de projetos de software livre, os **grupos especialistas** são os "*committers*". E têm o mesmo papel que os grupos especialistas propostos para a estratégia ColabSPI. Esses grupos têm o direito de alterar o OMM, subsidiando as novas versões do modelo. Assim como foi proposto para as grandes organizações, os grupos especialistas podem ser organizados de acordo com sua área de conhecimento. Por exemplo, um

grupo em particular pode deliberar sobre definição de métricas, enquanto outro decide sobre o processo de avaliação, ou, ainda, sobre um elemento de confiabilidade em particular. Monitorar e promover os canais de comunicação associados ao OMM (listas de mensagens, fóruns, etc.) são responsabilidades dos grupos especialistas.

Para a MPS em grandes organizações, a existência de dois grupos distintos (SEPG e grupos especialistas) mostrou-se uma prática interessante tanto do ponto de vista da evolução quanto da adequada utilização do processo. Para a evolução do OMM, essa distinção dependerá da quantidade de pessoas interessadas e, de fato, utilizando o modelo, já que a abrangência de projetos de software livre é imprevisível. À medida que o grupo de interessados cresce, a separação em grupos distintos torna-se mais oportuna e, nesse caso, o OPG focará na administração da evolução e definição de rumos do OMM, enquanto os grupos especialistas estarão mais focados nas alterações, nas publicações das novas versões e no controle de qualidade do que está sendo publicado. Incentivar contribuidores a tornarem-se membros de grupos especialistas é responsabilidade do OPG.

Cada **usuário** do OMM pode ser considerado um **contribuidor** em potencial. Eles podem contribuir: (i) reportando sua experiência no uso do modelo; (ii) propondo melhorias; (iii) encontrando e reportando inconsistências no modelo. Os contribuidores também podem ajudar traduzindo o modelo para outras línguas. Os usuários equivalem aos desenvolvedores na estrutura organizacional da ColabSPI para grandes organizações.

Por fim, **patrocinadores** do OMM podem oferecer recursos financeiros para a evolução do modelo e são equivalentes aos diretores em uma organização de grande porte.

6.3.2 O ciclo de evolução do OMM

Muito semelhantes aos passos para evolução do processo, propostos na ColabSPI, são os passos para evolução do OMM (Figura 57). Tal semelhança dispensa comentários adicionais acerca desses passos, que são detalhados em Malheiros e Maldonado (2009b).

Em relação ao processo de evolução dos diferentes modelos de maturidade disponíveis (ex.: CMMI e MR-MPS), o processo de evolução do OMM traz como inovação uma maior participação dos usuários do modelo na sua evolução, uma gestão transparente das propostas de melhoria ao modelo, e um canal de comunicação aberto e mantido pelos próprios usuários para informar sobre a evolução contínua do modelo.

No caso do MR-MPS, por exemplo, a sociedade SOFTEX é a coordenadora do programa e utiliza, para o seu desenvolvimento, a seguinte organização e atribuições (SOFTEX, 2010): coordenação geral; conselho de gestão do programa; equipe técnica do modelo; fórum de

credenciamento e controle; comissão ética do programa. A Tabela 12 apresenta uma comparação entre a estrutura para evolução do OMM e do MR-MPS.

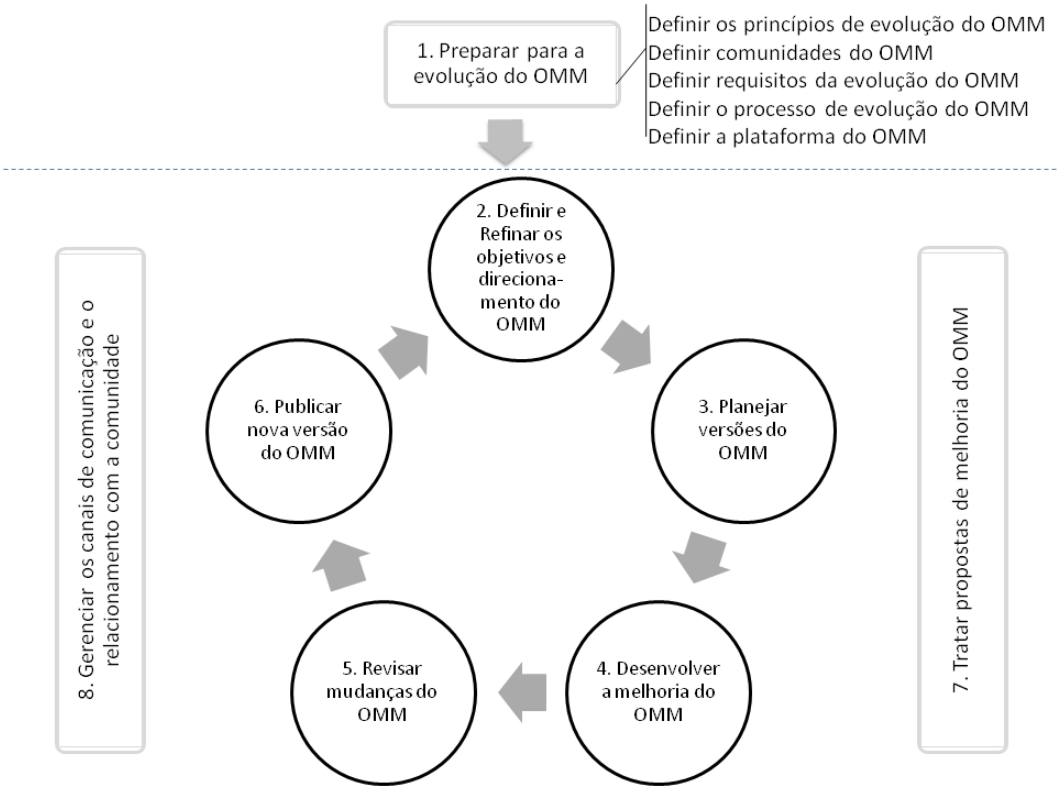


Figura 57: Processo de evolução do OMM (MALHEIROS e MALDONADO, 2009b)

Tabela 12: Comparação entre a estrutura para evolução do OMM e do MR-MPS

Comunidades para evoluir o OMM	Estrutura organizacional do MR-MPS
Patrocinador OMM	Coordenação geral
OPG	Conselho de gestão do programa
Grupos especialistas (*comitters*)	Equipe técnica do modelo
Ainda não definido	Fórum de Credenciamento e Controle
Ainda não definido	Comissão de Ética do Programa
Usuários	-

O patrocinador OMM desempenha parte das atividades previstas no MR-MPS para a coordenação geral, estando o patrocinador do OMM mais focado nas questões financeiras. O desenvolvimento de um código de ética não está previsto para o OMM, por esse motivo, não existe comissão de ética equivalente no OMM. Da mesma forma, ainda não está definida a forma como avaliadores do OMM serão capacitados, credenciados e monitorados. Apesar de existir um modelo para avaliação de aderência ao OMM, não está definido como o uso desse modelo de avaliação será sistematizado para que haja um caráter de certificação. Por isso, o fórum de credenciamento e controle não se aplica ao OMM, nesse momento. Por outro lado, como no

modelo de evolução do MR-MPS não está explicita uma participação direta da comunidade, o grupo usuários não aparece claramente na sua estrutura organizacional. Vale ressaltar que o WAMPS (Workshop Anual do MPS.BR) pode ser entendido como um mecanismo de participação da comunidade que pode contribuir para a evolução do MR-MPS.

6.3.3 A plataforma de evolução do OMM

Inspirada na infraestrutura da ColabSPI, a plataforma de evolução do OMM permite a integração de todas as informações relacionadas ao OMM. Essa infraestrutura (Figura 58) permite editar o modelo, incluindo funcionalidades de controle de versão e de comunicação e coordenação para administrar a evolução do modelo.

Figura 58: Infraestrutura para evoluir o OMM (MALHEIROS e MALDONADO, 2009b)

Diferentes soluções podem ser utilizadas para evoluir e disponibilizar o OMM. Na ColabSPI, duas soluções livres para documentação de processos foram consideradas: Atabaque e EPF Composer. As duas poderiam ser aplicadas para documentação do OMM em formato estruturado. O EPF Composer foi selecionado por ser uma solução mais difundida na comunidade de software livre, que manterá o modelo futuramente. Além disso, o EPF Composer permite a implementação do meta-modelo de processos de engenharia – SPEM (OMG, 2008) - que pode ser interessante para representar os componentes do OMM. Por exemplo, as práticas do OMM podem ser mapeadas para o elemento "práticas" do SPEM. Na Figura 59, é apresentada uma tela para edição da documentação do OMM utilizando a EPF Composer[56]. Na Figura 60, ilustra-se o OMM documentado por meio de EPF Composer.

A visualização em hipertexto por meio de um *site*, em contraposição com o formato de documento de modelos de maturidade como o CMMI ou o MR-MPS, foi avaliada como positiva e espera-se que ela aumente o interesse em conhecer melhor o modelo. Esse formato[56] foi muito útil para a elaboração do material de treinamento do OMM. A documentação em hipertexto em

[56] http://qualipso.icmc.usp.br/OMM/

um site pode facilitar a integração do modelo em si com o material de suporte e com as ferramentas que apóiam sua implementação. A árvore de navegação permite fácil acesso a todos os elementos da OMM *process suite* (quadro à esquerda da Figura 60).

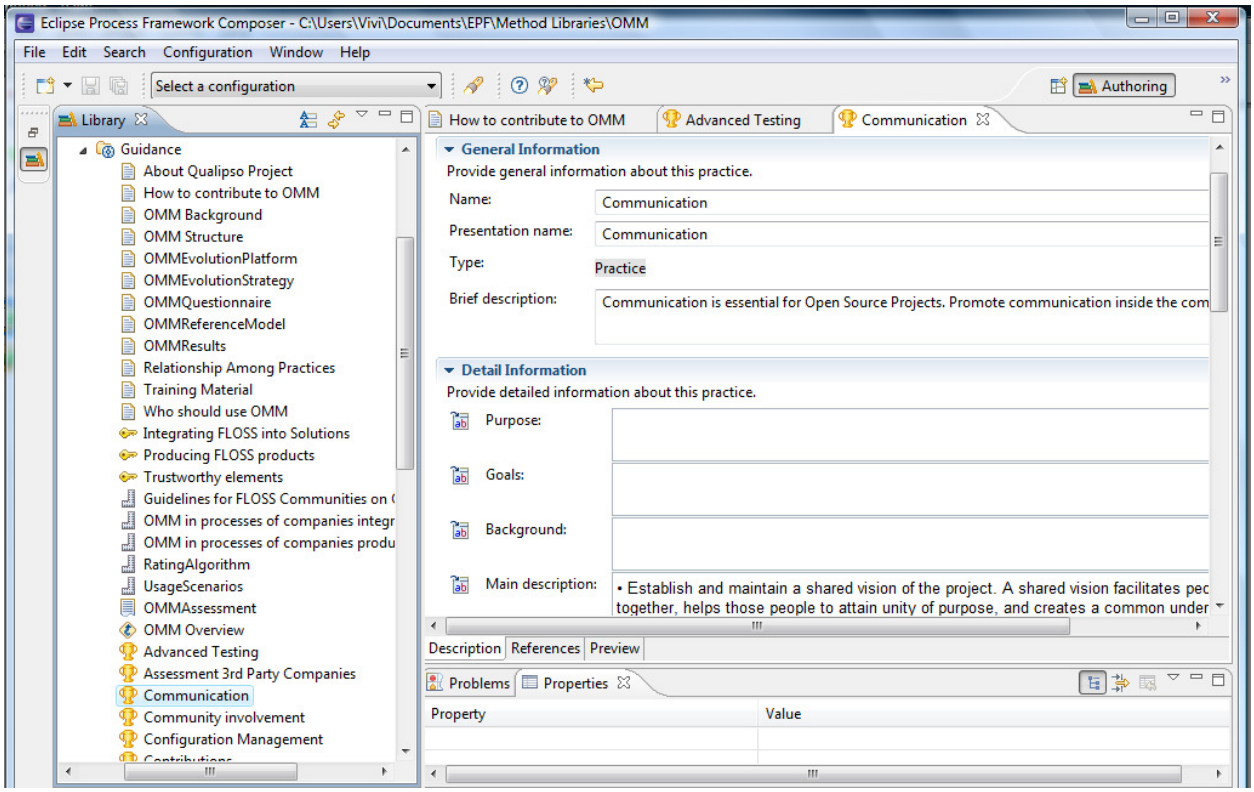


Figura 59: Tela de edição do OMM utilizando a EPF Composer

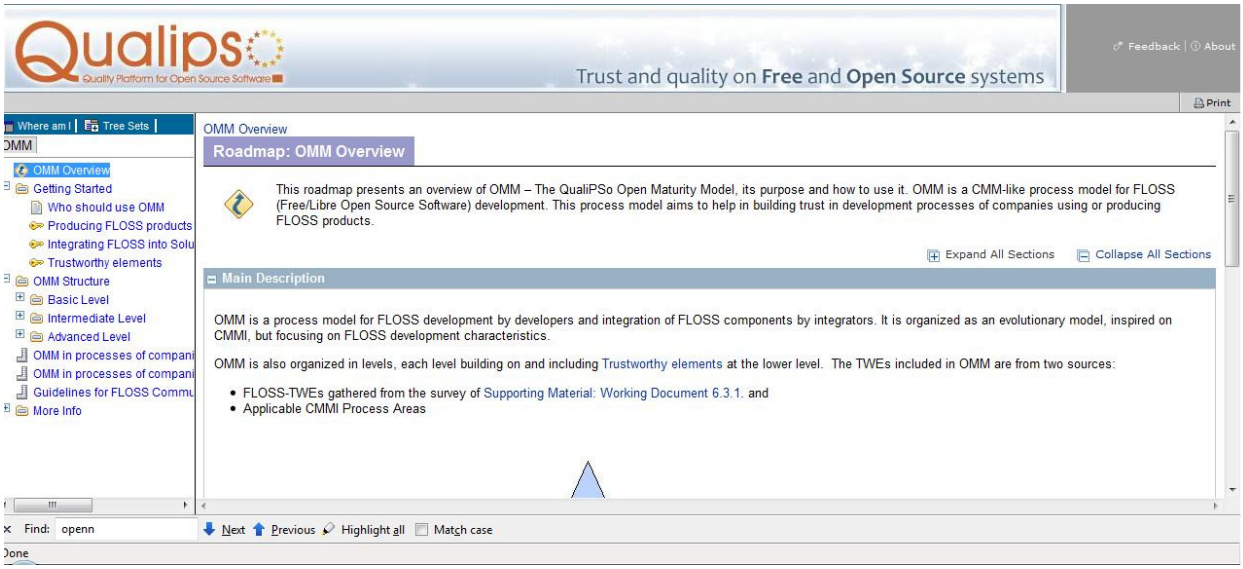


Figura 60: Visão do OMM em formato Web

Está em avaliação pelo grupo a utilização do EPF *Wiki* como ferramenta de apoio a coleta de sugestões de melhoria de processo e o uso de um ambiente de desenvolvimento colaborativo para a administração e comunicação do modelo.

6.3.4 Resultados preliminares da adaptação da ColabSPI para evoluir o OMM

Até o final do projeto Qualipso (previsão: final de 2010), a OMM *process suite* versão 1.0 será disponibilizada. Após o final do projeto, os três produtos da Atividade 6 deverão ser evoluídos e mantidos pelos Centros de Competência do Qualipso, por voluntários e por usuários do OMM na indústria.

Em princípio, o uso continuado de uma estratégia distribuída e colaborativa pode tornar a evolução do modelo mais efetiva. O foco primordial da ColabSPI é a evolução de processos. Apesar de, intuitivamente, ser possível visualizar os pontos em comum entre as duas necessidades de evolução (promover maior participação, viabilizar a comunicação e a colaboração entre os envolvidos), alguns aspectos podem ser particulares da evolução de processos ou da evolução de modelos. Por exemplo, já antevemos que os ciclos de evolução de um modelo devem ser maiores que os ciclos de evolução de um processo. Ainda assim, alguns resultados já podem ser observados.

O primeiro passo foi desenvolver um protótipo de um *site* do OMM com sua documentação estruturada, por meio do EPF Composer, e apresentá-lo para a equipe da atividade A6 do Qualipso. Foi realizada uma apresentação presencial da proposta. A equipe considerou a documentação do modelo por meio do EPF Composer como oportuna e interessante. O *site* foi apresentado para o coordenador geral do projeto Qualipso que registrou: "*at a first glance it is a very valuable work*". Além disso, a equipe da atividade A11 do Qualipso (Treinamento) declarou que o *site* foi essencial para compreender o modelo e gerar o seu material de treinamento.

Para confirmar a conveniência da documentação estruturada para os usuários em potencial do modelo, no questionário de avaliação do OMM (MALHEIROS et al., 2010a), foi incluída a questão: "*A web interface for OMM may help on reading and understanding the model more than the plain document version*", com o link para o OMM documentado com EPF Composer. Os participantes concordaram que o site ajudaria a ler e entender o OMM em 100% das respostas (70%, "*strongly agree*" e 30% "*agree*").

6.4 Considerações finais

Neste capítulo, foram apresentadas aplicações de mecanismos da ColabSPI em estudos experimentais na indústria e a possibilidade de aplicar estes mecanismos para evoluir o OMM.

A condução de estudos experimentais relacionados à MPS mostrou-se não trivial, dada a dificuldade de isolar varáveis e de estabelecer um relacionamento direto de causa (representado pelos mecanismos propostos) e efeito (representado pelos resultados alcançados).

Os estudos foram organizados em três grupos, de acordo com os mecanismos que eles possibilitaram exercitar: (i) à documentação do processo, (ii) ao tratamento de propostas de melhoria e (iii) às atividades de comunicação. Um quarto grupo analisou o uso de projetos piloto para a MPS. A aplicabilidade das estratégias e mecanimos para a MPS foi analisada por quantitativamente e qualitativamente. Duas soluções, Atabaque e Mantis-PMP, foram desenvolvidas e um ADC configurado para viabilizar a aplicação e observação das propostas.

Os estudos de cada grupo foram conduzidos em uma escala crescente de quantidade de informações observada, conforme proposto por Shull et al. (2001). A execução incremental dos estudos permitiu que os benefícios observados fossem paulatina e independentemente incorporados ao programa de MPS à medida que eram validados, ao mesmo tempo em que permitiu a definição incremental da estratégia e dos mecanismos da ColabSPI. Adicionalmente, o investimento de recursos para a execução de cada iniciativa de MPS cresceu apenas à medida que as idéias haviam sido validadas em um escopo mais específico. Vale ressaltar ainda que, à medida que as propostas foram sendo construídas e experimentadas, elas foram organizadas em publicações e submetidas à revisão da comunidade acadêmica. As contribuições obtidas a partir da revisão dos artigos submetidos e relatórios divulgados foram uma importante retroalimentação para a ColabSPI.

7- Conclusões

Neste capítulo são apresentadas as contribuições do trabalho para a melhoria de processo de software, bem como suas limitações. São também relacionados trabalhos futuros que podem dar continuidade à linha de pesquisa e potenciais novas contribuições.

A área de melhoria de processos de software tem estado em plena atividade, dadas as evidências de que a qualidade do processo pode influenciar significativamente na qualidade do produto final. Nos últimos anos, temas relacionados à MPS têm sido bastante discutidos, com o surgimento de modelo e guias e com a divulgação de lições aprendidas na implantação de programas de MPS na indústria. No entanto, apesar da expansão do conhecimento associado à área, poucos são os trabalhos que enfocam estratégias e mecanismos que indiquem *como* transformar as boas práticas apontadas nos modelos em melhorias aplicadas à indústria. Além disso, a tendência crescente de desenvolver de maneira distribuída aumenta a complexidade e a importância do processo de desenvolvimento de software, trazendo, ou reforçando, fatores que devem ser tratados. A MPS requer muito investimento e um tempo considerável (GOLDENSON e GIBSON, 2003). Por isso, contribuições que possam auxiliar sua efetividade são muito úteis.

O interesse em estudar o assunto surgiu da vivência da autora em um programa de MPS na indústria e da necessidade de encontrar soluções para implementar modelos de MPS, adaptando e implantando práticas recomendadas em uma organização de grande porte. Desde 2005, a autora é membro do SEPG corporativo do SERPRO, tendo assumido diferentes responsabilidades. A partir dessa vivência e da coleta de dados em relatórios, entrevistas, bancos de dados e reuniões foram identificados fatores que influenciavam a MPS, lições aprendidas e boas práticas (Capítulo 4).

Para subsidiar os trabalhos de pesquisa, além do estudo bibliográfico nas áreas de GC, MPS, DDS e desenvolvimento de SL, a autora trocou conhecimentos com especialistas dessas áreas participando em congressos e reuniões internacionais do projeto Qualipso. Em complemento, foi realizado um programa de intercâmbio na *University of Maryland, Baltimore County*, Estados Unidos da América, com a equipe da Profa. Dra. Carolyn Seaman.

7.1 Trabalho desenvolvido

A contribuição principal deste trabalho consiste em estratégias para apoiar uma MPS colaborativa e distribuída. Foram explorados dois contextos: (i) a MPS em uma organização de grande porte, com unidades distribuídas; e (ii) a MPS no desenvolvimento de software livre. Um ponto a ser ressaltado foi a oportunidade de desenvolver um trabalho de pesquisa aplicada, concentrado na interseção entre academia e indústria, no qual as lições aprendidas em cada uma das esferas (academia e indústria) serviram para subsidiar o trabalho desenvolvido na outra.

A ColabSPI, uma estratégia colaborativa e distribuída, foi desenvolvida para apoiar: (i) o tratamento de propostas de melhorias de processo pelas equipes de MPS, (ii) a evolução do processo; e (iii) a comunicação e colaboração entre desenvolvedores. Os objetivos estabelecidos foram atingidos, ainda que contando com algumas limitações (Seção 7.3). As contribuições geradas foram:

- Definição da ColabSPI, uma estratégia para a MPS colaborativa e distribuída. Essa estratégia combina práticas de GC com práticas do DDS e do desenvolvimento de SL. Os elementos definidos para a ColabSPI foram: (i) princípios, (ii) estrutura organizacional da MPS, (iii) etapas para a MPS e (iv) sua infraestrutura.
- Mapeamento e organização de fatores que podem influenciar a MPS. Para subsidiar a definição da ColabSPI, outras contribuições foram geradas, entre elas o mapeamento e organização dos fatores que podem influenciar a MPS. A partir de uma revisão sistemática, mais de 100 fatores relacionados à MPS foram identificados e, para melhor compreendê-los, um trabalho de mapeamento e organização dos fatores foi realizado. Observou-se que vários fatores críticos para a MPS dizem respeito à comunicação, à troca de conhecimento, à colaboração e, principalmente, ao grau de participação e motivação dos desenvolvedores em iniciativas de MPS. Os fatores foram organizados em cinco categorias e relações de causa e efeito entre os fatores encontrados foram exploradas. A organização dos fatores facilitou a definição das estratégias e distribuição das ações para a MPS, pois além de refletir os pontos que precisavam ser tratados, permitiu a separação das necessidades em: necessidades de uso, mais relacionadas à aderência ao processo (categoria Aderência), e necessidades ligadas à sua evolução (categoria Melhoria Contínua) (Capítulo 3). Além disso, a vivência da autora no programa de MPS do SERPRO (Capítulo 4) auxiliou no mapeamento de fatores para a MPS(Capítulo 3). Nesse sentido, outra contribuição foi a observação e a organização da experiência do SERPRO em MPS.

- Construção e configuração de ferramentas para implementar mecanismos da ColabSPI- Foram definidas e construídas as soluções Atabaque e Mantis-PMP, respectivamente para a documentação estruturada de processos e para o tratamento de PMPs. Adicionalmente, um ambiente de ADC foi adaptado para a MPS;
- Aplicação da estratégia para a MPS - Baseados em dados históricos e observações, ciclos de estudos foram realizados no SERPRO para investigar a aplicabilidade da estratégia e dos mecanismos propostos na indústria. Tais estudos foram conduzidos de maneira incremental, segundo níveis crescentes de informações obtidas. Os estudos experimentais apoiaram a análise dos três módulos da ColabSPI, incluindo o seu uso compartilhado por diferentes unidades de desenvolvimento e por essas unidades e o SEPG corporativo. Benefícios qualitativos e quantitativos foram observados e lições aprendidas coletadas. Por meio desses estudos, foi possível incorporar ao programa de MPS do SERPRO o conhecimento adquirido no decorrer deste trabalho de doutoramento. A aplicabilidade da estratégia para a MPS também foi considerada para evoluir o modelo OMM, estabelecendo a estratégia de evolução do OMM.

7.2 Outras contribuições dentro do trabalho de doutorado

Além das contribuições diretas desta tese (Seção 7.1), como parte do desenvolvimento das habilidades de pesquisa e do conhecimento da autora, foram conduzidos alguns trabalhos de pesquisa em paralelo:

- Uso de *Visual Text Mining* (VTM) para apoiar revisões sistemáticas - Ao conduzir pesquisas em Melhoria de Processos e Gestão de Conhecimentos a autora observou que as publicações nessas áreas, via de regra, não utilizam uma taxonomia que permita a recuperação precisa de trabalhos com palavras chave específicas. E, em complemento, palavras chave como "processo" retornam muitos artigos que não são relacionados ao tema. A autora propôs então a utilização de VTM (LOPES et al., 2007) como um instrumento de apoio a seleção de estudos em revisões sistemáticas. Esta proposta é apresentada em Malheiros et al. (2007) e deu origem a uma linha de pesquisa (um projeto de doutorado e um projeto de mestrado) que está explorando possibilidades adicionais de uso de VTM em revisões sistemáticas (FELIZARDO et al. 2010). Foi também considerado o uso de uma ferramenta incremental de mineração visual de conjuntos de documentos para auxiliar na atualização de revisões sistemáticas já realizadas (PINHO et al., 2010). A experiência com esse

trabalho foi importante para conduzir a revisão sistemática sobre os fatores críticos de sucesso da MPS;

- Revisão sistemática de gestão de conhecimento e testes de software - A autora conduziu uma revisão sistemática para identificar estudos sobre como as práticas de Gestão de Conhecimentos vinha sendo aplicada a área de testes de software. Mais uma vez, técnicas de VTM foram aplicadas na revisão. Esse trabalho aumentou a experiência da autora na condução de revisões sistemáticas. Os resultados desse trabalho estão sendo formatados em um artigo;
- Investigação de como melhorar um processo inspirado no processo unificado (abordagem tradicional) com princípios e metodologias ágeis. Durante os estudos sobre MPS, a incorporação de princípios das metodologias ágeis a um processo de desenvolvimento de software, inspirado no processo unificado (abordagem tradicional), foi investigada. Neste sentido, foram mapeados os pontos de intervenção no processo, que foi evoluído para uma nova versão. Além disso, as melhorias foram analisadas com base nos resultados de projetos-piloto (Seção 6.2.4). Tal análise realimentou o processo de desenvolvimento de software. Os resultados obtidos e as lições aprendidas foram organizados em um artigo, submetido ao *Journal of the Brazilian Computer Society* (JBCS) (MALHEIROS et al., 2010b);
- Contribuições para o OMM. Após o início dos trabalhos, surgiu a oportunidade de aplicar o conhecimento adquirido no contexto do projeto Qualipso, que se propôs a explorar a definição de um modelo para desenvolvimento de SL, nos moldes do CMMI. Em função dessa oportunidade, uma segunda frente de trabalho foi aberta, relacionada, agora, à aplicação das contribuições em MPS em outro contexto, o do desenvolvimento de SL (MALHEIROS e MALDONADO, 2009,b). A autora é uma das co-autoras do OMM (WITTMANN et al., 2009) e estruturou recomendações para um processo de desenvolvimento de software livre (Capítulo 4, MALHEIROS et al., 2009a);
- Definição, aplicação e análise de resultados de questionários para a avaliação do OMM. Por fim, ainda para o OMM, foram desenvolvidos e aplicados questionários para a avaliação do OMM. O planejamento e os resultados da avaliação do modelo estão disponíveis em Malheiros et al. (2010a).
- Definição de um processo livre para desenvolvimento de software. Mais recentemente, no contexto do projeto Demoiselle, capitaneado pelo SERPRO, iniciou-se a definição do processo Demoiselle (MALHEIROS et al. 2010c). Além do

processo em si, estão sendo discutidos aspectos relacionados à evolução de um processo livre e uma estratégia colaborativa e distribuída para MPS. O processo foi disponibilizado e está sendo mantido no Sourceforge, ambiente colaborativo difundido na comunidade de software livre. Esse ambiente implementa as estratégias propostas na ColabSPI, à exceção da Mantis-PMP que ainda não foi configurada externamente. A documentação do processo está versionada e escrita de forma estruturada, utilizando o EPF Composer para autoria e o SVN para gestão da configuração, e os mecanismos de comunicação foram criados.

7.3 Dificuldades e limitações do trabalho realizado

Uma das dificuldades enfrentadas durante a condução deste trabalho foi a necessidade de conciliar as características da pesquisa acadêmica com as características de um ambiente de produção industrial. Contudo, vale destacar que a possibilidade de explorar a integração entre esses dois ambientes foi justamente um dos motivadores do trabalho de doutorado e uma importante contribuição do mesmo.

Outra dificuldade está relacionada ao tema da pesquisa "Melhoria de Processo de Software". Como foi apontado no Capítulo 3, vários fatores críticos de sucesso da MPS estão relacionados a questões políticas, organizacionais e culturais. Essa característica da área impacta na condução de estudos experimentais para caracterizá-la, avaliá-la ou e melhorá-la. Durante a execução deste trabalho, a estrutura organizacional da empresa observada foi alterada duas vezes, incluindo a formação do SEPG corporativo. A cada alteração, as prioridades do programa de MPS são revisitadas e estudos precisam ser redirecionados ou postergados. Tal dificuldade deve ser considerada para estudos experimentais que extrapolem a fronteira das questões técnicas da Engenharia de Software e que, de alguma forma, estejam também relacionados às questões sociais das organizações de desenvolvimento de software. Os impactos dessas alterações nos estudos conduzidos podem ser: reavaliação de cronograma; readequação dos estudos experimentais para o novo contexto; e revisão dos mecanismos propostos, visto que a definição dos mesmos depende não apenas de decisões técnicas, mas também de decisões organizacionais. Mesmo com essa dificuldade, no entanto, é importante que tais estudos sejam conduzidos no ambiente industrial, já que seria ainda mais difícil avaliar as questões citadas em laboratórios ou com estudantes no lugar de profissionais.

Ainda no tocante ao tema de pesquisa escolhido, a natureza multidisciplinar da proposta de projeto trouxe a necessidade de investigação e estudo de uma diversidade de temas e linhas de

pesquisa, além da investigação dos estudos na interseção entre essas áreas, expandindo consideravelmente o referencial teórico a ser explorado. Adicionalmente, diferentes linhas de pesquisa estão surgindo com possibilidades de contribuição para a MPS. Algumas dessas linhas não foram consideradas no escopo deste trabalho, a exemplo de: (i) otimização quantitativa de desempenho de processos; (ii) estudos relacionados à automação de processos e (iii) notação BPMN (*Business Process Modeling Notation*) para modelagem de processos.

Em relação às limitações deste trabalho, os estudos experimentais da MPS na indústria foram conduzidos em uma mesma empresa de grande porte. Para avaliar se os resultados encontrados podem ser generalizados para outros contextos industriais, foi realizada uma revisão sistemática para identificação dos fatores críticos de sucesso de MPS reportados em outras experiências. A semelhança entre muitos dos fatores críticos encontrados é um indício de que os resultados podem ser aproveitados em outros contextos. Ainda assim, a realização de experimentos em uma única empresa, ainda que em momentos distintos e sob diferentes configurações organizacionais, deve ser considerada como uma fonte de viés para os resultados. Apesar de vários estudos práticos terem sido realizados, não foram conduzidos experimentos controlados com separação de variáveis e controle de cada etapa do estudo.

A proposta apresentada deixou de abordar alguns aspectos e técnicas que também poderiam contribuir para a MPS, a saber: (i) recursos da gestão de competências, que permitiriam compreender as competências organizacionais críticas para a MPS e o mapeamento de quem sabe o quê em relação a MPS; (ii) como tratar a MPS para processos automatizados, o foco do trabalho foi na evolução de processos documentados; e não na otimização de processos automatizados. (iii) validação da aplicação da estratégia de evolução para o OMM ou outros modelos de maturidade; (iv) aplicação da ColabSPI para a MPS de um processo utilizado por várias organizações de pequeno ou médio porte. É possível que a estratégia seja útil se for possível estabelecer um paralelo entre cada organização de pequeno e médio porte e as unidades organizacionais de uma organização de grande porte; (v) como avaliar a usabilidade de processos, para subsidiar decisões de como melhor documentá-los.

7.4 Perspectivas de trabalhos futuros

Dando continuidade às atividades conduzidas durante este trabalho de doutorado, podem-se destacar como principais perspectivas futuras de pesquisa as seguintes linhas de atuação:

- Aplicar estratégias de MPS propostas neste trabalho em outros contextos. Como um dos principais trabalhos futuros, tem-se a aplicação (e adequação) da ColabSPI para

outros contextos, entre eles, o de outras organizações de desenvolvimento de software. Ademais, o uso de estratégias de MPS para a evolução de processos livres de software pode ser explorado para aumentar a qualidade desses processos. Devido às características de trabalho distribuído e colaborativo, típicas de equipes de desenvolvimento de software livre, o uso da ColabSPI pode facilitar a evolução de processos livres e facilitar e contribuir para o uso adequado dos processos propostos nas fábricas de experiência do Qualipso. Outro possível contexto de aplicação é a evolução de processos que sejam compartilhados por organizações de médio ou pequeno porte. Empresas podem compartilhar um conjunto de ativos de processo e evoluí-lo em conjunto, a exemplo do que ocorre com o programa MPS.BR;

- Investigar estratégias e mecanismos para gestão de competências de MPS. Uma das alternativas para promover a troca de conhecimento é utilizar técnicas de gestão de competências, com a representação em mapas de competência de quem sabe determinado assunto, quem mais contribui para a MPS ou quem mais esclarece determinado tipo de dúvida. Para desenvolver tais mapas de competência, um trabalho preliminar pode ser o mapeamento e organização do conhecimento de uma determinada área;
- Investigar como tratar a MPS de forma colaborativa para processos automatizados de software. Os aspectos de MPS discutidos e validados nesta tese se referem a processos de software documentados. Pode ser interessante investigar aspectos da evolução de processos automatizados de software e de ambientes de engenharia de software orientados a processos;
- Investigar e avaliar a usabilidade de processos de software. Durante a condução deste trabalho, surgiram dúvidas sobre qual seria a melhor forma de documentar e disponibilizar todos os elementos de um processo de software e como avaliar e melhorar a usabilidade destes processos. O tema usabilidade tem crescido em importância nos estudos em Engenharia de Software. Ainda assim, não foram encontrados estudos abrangentes sobre a usabilidade de processos de software;
- Avaliar a adequação da ColabSPI para evoluir o OMM. Acompanhar a evolução do OMM com o uso do modelo de evolução adaptado de ColabSPI pela comunidade de software livre; e
- Evoluir de forma distribuída e colaborativa um processo livre para desenvolvimento de software. O processo Demoiselle, cuja primeira versão foi desenvolvida no

SERPRO, foi disponibilizado para a comunidade em um ambiente de acesso público, Sourceforge, permitindo a evolução distribuída e colaborativa de um processo livre para desenvolvimento de software que pode ser lido, utilizado e adaptado por todos os interessados.

7.5 Produção científica

Como publicações e apresentações resultantes das atividades realizadas durante o trabalho de doutorado, têm-se, da mais recente para a mais antiga:

- MALHEIROS, V. SEAMAN, C. MALDONADO. *An Approach for Collaborative and Distributed Software Process Improvement (SPI)*. Versão estendida do artigo para o WDDS 2009. Aceito para publicação no Infocomp - *Journal of Computer Science*.
- MALHEIROS, V.; HÖHN, E.; MESSIAS, R., MALDONADO, J.C.; PETRINJA, E.; SILLITI, A., WITTMAN, M.; ORTEGA, F.; RUFFATI, G. *WD6.5.4 - Survey Results Version 1*. Relatório Técnico Projeto Qualipso. Janeiro, 2010.
- MALHEIROS, V. SEAMAN, C. MALDONADO, J. *An Approach for Collaborative and Distributed Software Process Improvement (SPI).* III Workshop de Desenvolvimento Distribuído de Software (WDDS), 2009.
- WITTMANN, M., NAMBAKAM, R., PETRINJA, E., ORTEGA, F., RUFFATI, G., OLTOLINA, S., MALHEIROS, V. *Working Document WD6.3.1 - CMM-like model for OSS, version 1*. Relatório Técnico Projeto Qualipso. Dezembro, 2009.
- MALHEIROS, V. MALDONADO, J.C. *Qualipso Project: Open Process for Evolution, Maintenance and Administration of OMM*. Relatório Técnico RT-347, Universidade São Paulo, 2009.
- MALHEIROS, V. *Quality of FLOSS development: Yes, OMM can!* Palestra selecionada para o Fórum Internacional de Software Livre, 2009. Disponível em: http://ccsl.ime.usp.br/files/FISL%20-%20Quality%20of%20FLOSS%20development%20-%20Yes%20OMM%20can.pdf
- MALHEIROS, V., HÖHN, E., MALDONADO, J.C. *Qualipso Project: Quality Recommendations for FLOSS development processes.* Relatório Técnico RT-335, Universidade São Paulo, 2009. Disponível em: http://www.icmc.sc.usp.br/~biblio/BIBLIOTECA/rel_tec/RT_335.pdf

- MALHEIROS, V.; PAIM, F. R.; MENDONÇA, M. *Continuous Process Improvement at a Large Software Organization. Software Process Improvement and Practice*, v. 13, p. 1-16, 2008.
- MITCHELL, S. SAMPATH, S. MALHEIROS, V. *Leveraging Knowledge Management during Software Product Development*. Consortium on Computing Sciences in Colleges East (CCSC-E'08), Frederick, USA, October 2008.
- MALHEIROS, V; REHEM, S; MALDONADO, J. C. Atabaque: uma contribuição de sucesso na evolução de processos. Em: Anais do VII Simpósio Brasileiro de Qualidade de Software (SBQS), Florianópolis, 2008.
- MALHEIROS, V.; HÖHN, E.; PINHO, R.; MENDONÇA, M.; MALDONADO, J. C. *A Visual Text Mining approach for Systematic Reviews.* Em: *International Symposium on Empirical Software Engineering and Measurement* (ESEM), 2007, Madrid. *Proceedings of the 1st ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (ESEM), 2007.
- MALHEIROS, V., CUNHA L., REHEM, S. Mantis-PMP: uma ferramenta livre para gestão de mudanças em processos. Em: Congresso SERPRO de Tecnologia e Gestão Aplicadas a Serviços Públicos - ConSerpro 2007, Porto Alegre 2007.
- MALHEIROS, V.; PAIM, F.; GUZZO, H.; MENDONÇA, M. Uma Abordagem para Melhoria Contínua do Processo de Desenvolvimento de Software. Simpósio Brasileiro de Qualidade de Software 2006 (SBQS), Vila Velha – ES, 2006.

Recentemente foi submetido para publicação o trabalho:

- MALHEIROS, V.; PAIM, F.; MENDONÇA, M.; MALDONADO, J.C. *Software Process in a Large Organization: Can we get agile?* Submetido ao *Journal of the Brazilian Computer society* (JBCS), aguardando avaliação.

Os seguintes artigos e relatórios estão também em estágio avançado de elaboração:

- MALHEIROS, V.; MITCHELL S.; SEAMAN,C.; SAMPATH S.; MALDONADO, J. C. *Knowledge Management in Software Testing: A systematic review*. Artigo em elaboração.
- HÖHN, E.; MALHEIROS, V.; MALDONADO, J. C.; FORTES, R.; PETRINJA, E. *Working Document 6.5.3 Surveys Design.* Relatório técnico em elaboração.

Referências bibliográficas

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 9241-11 Requisitos ergonômicos para trabalho de escritórios com computadores. Parte 11 – Orientações sobre usabilidade. Rio de Janeiro: ABNT, 2002.

ABRAN, A.; MOORE, J. *Guide to the Software-Engineering body of knowledge, trial version (version 1.0)*. IEEE Computer Society, Los Alamitos, 2001.

ABRAHAMSSON, P. *Commitment development in software process improvement: critical misconceptions*. Em: Anais da *International Conference on Software Engineering* – ICSE 2000, Toronto, Canadá, 2000.

ABRAHAMSSON, P.; WARSTA, J.; SIPONEN, M.; RONKAINEN, J. *New directions on agile methods: a comparative analysis*. Em: *International Conference on Software Engineering*, 2003. DOI: 10.1109/ICSE.2003.1201204.

ACUÑA, S.T.; JURISTO, N. *Software Process Modelling*. Springer Verlag, 2005.

ACUÑA, S.; FERRÉ, X. *Software Process Modelling*. Acesso em: 18/03/2010. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.5438&rep=rep1&type=pdf..

ACHATZ, R. *Optimization of software development.* Em: *Proceedings of the 28th international Conference on Software Engineering* (Shanghai, China, May 20 - 28, 2006). ICSE '06. ACM, New York, NY, 30-30, 2006. DOI= http://doi.acm.org/10.1145/1134285.1134289

AHONEN, J.; LINTINEN, H.; TASNINEN, S.-K. *Improving the Reuse Process is Based on Understanding the Business and the Products: Four Case Studies.* Em: *Lecture Notes in Computer Science: Product Focused Software Process Improvement*, Springer-Verlag, 2002.

AMBLER, S. NALBONE, J., VIZDOS, M. *The Enterprise Unified Process: Extending the Rational Unified Process*. Prentice Hall, 2005.

APPLETON, B. *Patterns for Conducting Process Improvement*. Em: *The 4th Pattern Languages of Programming Conference* (PLoP), 1997. Disponível em: http://www.cmcrossroads.com/bradapp/docs/i-spi/plop97.html. Acesso em: 09/02/2010.

ARAUJO, M. Automatização do Processo de Desenvolvimento de Software nos Ambientes Instanciados pela Estação TABA. Dissertação de Mestrado, orientador: Guilherme Horta Travassos, COPPE, UFRJ, 1998.

ARAUJO, R. BORGES, M. *The Role of Collaborative Support to Promote Participation and Commitment in Software Development Teams*. *Software Process Improvement and Practice*, v. 12, p. 229–246, Wiley Interscience, 2007.

ARBAOUI, S.; DERMIANE, J.; OQUENDO, F. *Comparative review of process centered Software Engineering Environments*". Software Engineering 14, p.311-340.

ASSUNDI, J. *Software engineering lessons from open source projects. Making Sense of the Bazaar*. Em: *Proceedings of the 1st Workshop on Open Source Software Engineering*. Feller, J., Fitzgerald, B. & van der Hoek, A.(eds), 2001.

BADDOO, N., HALL, T. *Motivators of Software Process Improvement: an analysis of practitioners' view*. *Journal of System and Software*, 2002.

BADDOO, N., HALL T. *De-motivators of software process improvement: an analysis of practitioners' views*. *Journal of Systems and Software*, 2003. 66(1): 23–33.

BADDOO, N., HALL T. O'KEEFFE, C. *Using Multi Dimensional Scaling to Analyse Software Engineers' De-motivators for SPI*. *Software Process Improvement Practice*, vol. 12, 2007.

BANDINELLI, S.; BRAGA, M.; FUGGETTA, A. LAVAZZA, L. *The architecture of the SPADE-1 Process-Centered SEE.* Lecture Notes in Computer Science, v.772, 1994.

BARBOSA, E. Uma Contribuição ao Processo de Desenvolvimento e Modelagem de Módulos Educacionais. Tese de Doutoramento, Instituto de Ciências Matemáticas e de Computação – ICMC-USP, São Carlos, SP, 2004.

BARBOSA, E.; ADRIANO, C.; MALDONADO, J.; RICARTE, I.; JINO, M. *Fostering Theoretical, Empirical and Tool Specific Knowledge in a Software Testing Learning Scenario*. Em: *Proceedings of the International Conference on Engineering and Computer Education* (ICECE 00), p. 1–4, 2000.

BARBOSA, E.; NAKAGAWA, E.; MALDONADO, J. C. *Towards the establishment of an ontology of software testing*. Em: 18th *International Conference on Software Engineering & Knowledge Engineering*, 2006.

BASILI, V.; CALDIERA, G.; ROMBACH, H. *The goal question metric approach. Encyclopedia of software engineering*, v. 1, p. 528-532. John Wiley & Sons 1994. 1994a.

BASILI, V.R.; CALDIERA, G.; ROMBACH, H.D. *Experience Factory.* Em: *Encyclopedia of Software Engineering*, v. 1, p. 469—476, 1994b.

BASILI, V. CALDIERA, G. *Improve Software Quality by Reusing Knowledge and Experience.* Sloan Management Review, v.37, p. 55-64, 1995.

BASILI, V. *The Role of Experimentation in Software Engineering: Past, Current, and Future*. Em: Anais da *18th International Conference on Software Engineering*, 1996.

BASILI, V. HEIDRICH, J. LINDVALL, M. MUNCH, J. REGARDIE, M. TRENDOWICZ, A. *GQM+ Strategies - Aligning Business Strategies with Software Measurement*. Em: *First International Symposium on Empirical Software Engineering and Measurement* (ESEM), 2007. ISBN: 978-0-7695-2886-1

BECK, K. *Embracing change with extreme programming*. Em: IEEE Magazine, v. 32, p. 70–77, 1999. DOI= http://dx.doi.org/10.1109/2.796139

BECK, K.; BOEHM, B. *Agility through discipline: a debate*. IEEE Computer, v. 36 n. 6, p. 44-46 Jun. 2003. DOI=10.1109/MC.2003.1204374

BEECHAM, S. HALL, T., RAINER, A. *Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis*. Em: *Empirical Software Engineering*, v. 8, p.7–42, 2003.

BERGLUND, A. *Extensible Stylesheet Language (XSL) Version 1.1. World Wide Web Consortium Working Draft 16*, 2004. Disponível em: http://www.w3.org/TR/2004/WD-xsl11-20041216/ Acesso em: 03/11/2009.

BERTOLLO, G. SEGRINI, B. FALBO, R. Definição de processos de software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias. Em: V Simpósio Brasileiro de Qualidade de Software, Anais, 2006.

BIRK A., SURMANN D., ALTHOFF K. *Applications of knowledge acquisition in experimental software engineering*. Em: *11th European Workshop on Knowledge Acquisition, Modeling and Management*, p. 67–84, 1999.

BJORNSON, F. *Knowledge Management in Software Process Improvement*. Tese de Doutoramento. *Norwegian University of Science and Technology*, 2007 (Apêndice A).

BOEGH, J.; DEPANFILIS, S.; KITCHENHAM, B.; PASQUINI, A. *A method for software quality planning, control, and evaluation. Software.* Em: IEEE, v. 16, n. 2, p. 69–77, 1999.

BOEHM, B. *Get ready for Agile Methods with care*. Em: IEEE Computer 35, 1, 64-69, 2002 DOI= http://dx.doi.org/10.1109/2.976920.

BOEHM,B.;TURNER,R. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley, 2004.

BOOCH G.; BROWN, A.W. *Collaborative Development Environments*. Rational Software Corporation, 2002

BORGES, L. S.; FALBO, R. A. Gerência de Conhecimento sobre Processos de Software. Em: Anais do VIII Workshop de Qualidade de Software, 2001, p. 27–38.

BRAGA, R.; MASIEIRO, P. Um processo para construção e instanciação de *frameworks* baseados em uma linguagem de padrões para um domínio específico. Tese de Doutoramento, ICMC/USP, 2002.

BRAGA, R.; WERNER, C.; MATTOSO, M. *Odyssey: a reuse environment based on domain models*. Em: *IEEE Symposium on Application-Specific Systems and Software Engineering and Technology* (*ASSET'99. Proceedings*), p. 50–57, 1999.

BRAY, T.; PAOLI, J.; SPERBERG-McQUEEN, C.; MALER, E. YERGEAU, F. *Extensible Markup Language (XML) 1.0 (Fifth Edition). World Wide Web Consortium Recommendation 26*. Nov. 2008. Disponível em: <http://www.w3.org/TR/REC-xml>. Acesso em: 03/11/2009

BROTHERS, L.; SEMBUGAMOORTHY, V.; MULLER, M. *ICICLE: groupware for code inspection*. Em: Anais da ACM Conference on Computer-Supported Cooperative Work (Los Angeles, California, United States, 1990). CSCW '90. ACM, New York, NY, 169-181. DOI= http://doi.acm.org/10.1145/99332.99353

CAGNIN, M. Uma contribuição para a reengenharia de software baseada em linguagens de padrões e *frameworks*. Tese de Doutoramento, Instituto de Ciências Matemáticas e de Computação – ICMC-USP, São Carlos, SP, 2005.

CAGNIN, M.; MALDONADO, J.; CHAN, A.; PENTEADO, R.; GERMANO, F. Reúso na Atividade de Teste para Reduzir Custo e Esforço de VV&T no Desenvolvimento e na Reengenharia de Software. XVIII Simpósio Brasileiro de Engenharia de Software, 2004.

CAMPOS, F.; ALBUQUERQUE, A.; ANDRADE, J.; SILVA Filho, R.; ROCHA, A. Abordagem em Níveis para Avaliação e Melhoria de Processo de Software. Em: Anais do Simpósio Brasileiro de Qualidade de Software, p. 100 –114, Vitória, ES, 2006.

CARMEL, E. *Global Software Teams – Collaborating Across Borders and Time Zones*. EUA: Prentice Hall, 1999.

CARNEIRO, C. *Frameworks* de Aplicações Orientadas a Objetos - Uma Abordagem Iterativa e Incremental. Dissertação de Mestrado, Mestrado em Redes de Computadores, Universidade Salvador, Salvador, 2003.

CARVALHO, I.; MENDES, S. P.; VERAS, V. Gestão do Conhecimento: Uma estratégia empresarial 1ª. ed. J.J Gráfica e Comunicações Ltda., Curitiba, 2006.

CHAU, T.; MAURER, F.; MELNIK, G. *Knowledge Sharing: Agile Methods vs. Tayloristic Methods*. Em: *Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE'03), 2003.

CLEMENTS, P.; NORTHROP, L. *Software product lines*. Addison-Wesley Boston, 2002.

COHN, M.; FORD, D. *Introducing an agile process to an organization*. IEEE Computer, v. 36, n. 6, 74-78. Junho, 2003. DOI= http://dx.doi.org/10.1109/MC.2003.1204378

COLEMAN, D. KHANNA, R. *Groupware technology and applications: an overview of groupware.* Prentice Hall International (UK) Ltd., Hertfordshire, UK, 3-41, 1995

COLEMAN, G. O'CONNOR, R. *Investigating software process in practice: A grounded theory perspective*. Em: *The Journal of Systems and Software,* v. 81 p. 772–784, 2008

CONRADI, R.; FUGGETTA, A. *Improving software process improvement*. IEEE Software, 2002. DOI= http://dx.doi.org/10.1109/MS.2002.1020295

CUBRANIC, D.; BOOTH, K. *Coordinating open-source software development.* Em: *Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1999. (WET ICE'99), *IEEE 8th International Workshops*, p. 61–66, 1999.

CUNHA, L. Avaliação de ferramentas de *bugtracking* observando conceitos do CMM e teoria de testes. Monografia apresentada ao Curso de Especialização Avançada em Sistemas Distribuídos. Orientador: Prof. Dr. João Gualberto Rizzo Araújo. UFBA, julho, 2005.

CURTIS, B., KELLNER, M., OVER, J. *Process Modelling*. Em: Communications of the ACM, ISSN:0001-0782, v. 35, n.9, p. 75-90, 1992.

DEMING, W. *Out of the crisis*. Cambridge: MIT Center for Advanced Engineering Study, 1986.

DESOUZA, K. C. *Barriers to effective use of knowledge management systems in software engineering*. Communications of ACM, v. 46, n. 1, p. 99–101, 2003.

DINKELACKER, J., GARG, P. K., MILLER, R., NELSON, D. 2002. *Progressive open source*. Em: *Proceedings of the 24th international Conference on Software Engineering* (Orlando, Florida, May 19 - 25, 2002). ICSE '02. ACM, New York, NY, 177-184. DOI= http://doi.acm.org/10.1145/581339.581363.

DUARTE, K.; FALBO, R. Uma Ontologia de Qualidade de Software. VII Workshop de Qualidade de Software, 2000.

DYBA, T. *An Instrument for Measuring the Key Factors of Success in Software Process Improvement*. Empirical Software Engineering, 5, 357–390, Boston, 2000

DYBA, T. *Enabling Software Process Improvement: An Investigation of the importance of Organizational Issues.* Tese de Doutoramento, Department of Computer and Information Science, Norwegian University of Science and Technology, 2001.

DYBA, T.; DINGSOYR, T. *Empirical studies of agile software development: A systematic review*. Em: Information and Software Technology, v. 50, p. 833-859, 2008.

EARL, M. *Knowledge Management Strategies: Toward a Taxonomy*. *Journal of Management Information Systems*, v. 18 (1), p. 215-233, 2001.

ECLIPSE FOUNDATION. *About us*. 2007a. Disponível em http://www.eclipse.org/org/ Acesso em: 06.01.2010

ECLIPSE FOUNDATION. *Agile component*. 2007b. Disponível em http://www.eclipse.org/epf/agile_component/agile_objectives.php. Acesso em 03.03.2010.

ECLIPSE FOUNDATION. *Eclipse newcomers faq*. 2007c. Disponível em http://www.eclipse.org/home/newcomers.php. Acesso em 02.02.2007.

ECLIPSE FOUNDATION. *Eclipse process framework project (EPF)*. 2007d. Disponível em http://www.eclipse.org/epf/general/description.php. Acesso em 2009.

ECLIPSE FOUNDATION. *Initial vision of openup/mdd*. 2007e. Disponível em https://bugs.eclipse.org/bugs/attachment.cgi?id=39648. Acesso em 02.02.2007.

ECLIPSE FOUNDATION. *Openup component*. 2007f. Disponível em http://www.eclipse.org/epf/openup_component/openup_index.php. Acesso em 02.02.2007

ECLIPSE FOUNDATION. *Tool component vision*. 2007g. Disponível em http://www.eclipse.org/epf/tool_component/tool_vision.php. Acesso em 02.02.2007.

EL_EMAN, K., SMITH, B., FUSARO, P. *Success factors and barriers in software process improvement*. In: R. Messnarz and C. Tully, IEEE Computer Society Press, Silver Spring,1999.

EL-EMAM, K., GOLDENSON, D., MCCURLEY, HERBSLEB, J. *Modeling the Likehood of Software Process Improvement: An Exploratory Study*. Empirical Software Engineering, vol. 6, p. 207-229. Kluwer Academis Publisher, 2001.

ELLIS, C., GIBBS S., REIN, G. *GROUPWARE: some issues and experiences*. Communications of the ACM v. 34, n. 1, p. 39–58, 1991.

ERDOGMUS, H. *Essentials of Software Process*. IEEE Software v. 25, n. 4, p. 4-7, 2008. DOI= http://dx.doi.org/10.1109/MS.2008.87

FAYAD, M.; SCHMIDT, D. *Object-Oriented Applications Frameworks*. *Communications of the ACM*. *Special Issue on Object-Oriented Applications Frameworks*, v. 40, n. 10, 1997.

FALBO, R. Integração de Conhecimento em um Ambiente de Desenvolvimento de Software. Tese de doutorado, COPPE, UFRJ. Dezembro,1998.

FALBO, R.; NATALI, A.; MIAN, P.; BERTOLLO, G.; RUY, F. ODE: *Ontology-based software Development Environment.* Em: *Proceedings of the IX Argentine Congress on Computer Science* (CACIC˝S2003), La Plata, Argentina, p. 1124–1135, 2003.

FALBO, R.; RUY, F.; BERTOLLO, G.; D.F., T. *Learning How to Manage Risks Using Organizational Knowledge.* Em: *6th International Workshop on Learning Software Organization*, LSO'2004, 2004.

FAPESP - Fundação de Amparo a Pesquisa do Estado de São Paulo. Incubadora virtual. 2006. Disponível em http://incubadora.fapesp.br/ . Acesso em 15/06/2006.

FELIZARDO, K.; NAKAGAWA, E.; FEITOSA, D.; MINGHIM, R.; MALDONADO, J.C. *An Approach Based on Visual Text Mining to Support Categorization and Classification in the Systematic Mapping*. *EASE 2010 - Evaluation and Assessment in Software Engineering* - Keele University, Staffordshire, United Kingdom, 2010.

FERREIRA, A., SANTOS, G., CERQUEIRA, R., MONTONI, M., BARRETO, A., SOARES BARRETO, A. O., ROCHA, A. R. *Applying ISO 9001: 2000, MPS.BR and CMMI to Achieve Software Process Maturity: BL Informatica's Pathway*. Em: *Proceedings of the 29th international Conference on Software Engineering*. IEEE Computer Society, Washington, DC, 642-651. Maio, 2007 DOI= http://dx.doi.org/10.1109/ICSE.2007.15

FIORINI, S. Arquitetura para Reutilização de Processos de Software. Tese de Doutoramento, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, orientador: Júlio César Prado Leite, 2001.

FLORAC, W., PARK, R., CARLETON, A. *Practical Software Measurement: Measuring for Process Management and Improvement*. Guidebook CMU/SEI-97-HB-003. *Software Engineering Institute*, Carnegie Mellon University, Pittsburgh, PA 15213. 1997.

FREE SOFTWARE FOUNDATION. *Free Software Definition*. 2009 Disponível em http://www.gnu.org/philosophy/free-sw.html. Acesso em 27.03.2010.

FREE SOFTWARE FOUNDATION. *Various licenses and comments about them*. 2008. Disponível em http://www.gnu.org/licenses/license-list.html. Acesso em 01.03.2010.

FREITAS, A.V. APSEE-Global: um Modelo de Gerência de Processos Distribuídos de Software. Dissertação de Mestrado. Porto Alegre: Programa de Pós-Graduação em Computação, 2005.

FUGGETTA, *A. Software process: a roadmap*. Em: *Proceedings of the Conference on The Future of Software engineering*, p. 25–34, 2000.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Boston, USA, 1995.

GENVIGIR, E. SANT'ANNA, N. FILHO, L., JUNIOR, M., CASILLO, B. Modelando processos de software através do UPM – Modelo de Processo Unificado. Em: CITS 2003, Congresso Internacional de Tecnologia de Software, Curitiba, Paraná, p. 28-41, 2003.

GEROSA, M.; FUKS, H.; LUCENA, J. Suporte à Percepção em Ambientes de Aprendizagem Colaborativa. Revista Brasileira de Informática na Educação, v.11 n. 2, 2003.

GIMENES, I. Introdução aos Processos de Software . Disponível em: http://www.cin.ufpe.br/~gamr/FAFICA/Desenvolvimento%20de%20sistemas/Aula%203%20-%20Processos.ppt. Acesso em 25/02/2010.

GOLDENSON, D. R.; HERBSLEB, J. *After the Appraisal: A systematic Survey of Process Improvement, its benefits and Factors that influence Success*. Relatório Técnico, CMU/SEI-95-TR-009, Carnegie Mellon University, Software Engineering Institute, SEI, 1995.

GOLDENSON, D.R., GIBSON, D.L. *Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results*, SEI Special Report, CMU/SEI-2003- SR-009, Out., 2003.

GREMBA, J.; MYERS, C. *The IDEAL Model: A Practical Guide for Improvement.* Em: *Software Engineering Institute (SEI) Publication*, Bridge, v. 3, 1997. Disponível em http://www.sei.cmu.edu/ideal/ideal.bridge.html

GRUDIN, J. *Computer-Supported Cooperative Work: History and Focus*. Computer 27, 5 (May. 1994), 19-26, 1994 DOI= http://dx.doi.org/10.1109/2.291294

GRUNBACHER, P. HALLING, M. BIFFL, S. *An empirical study on groupware support for software inspection meetings*. Em: Anais da 18a. *IEEE International Conference on Automated Software Engineering* (ASE'03), 2003.

GURBANI, V., GAVERT, A., HERBSLEB, J. *A Case Study of a Corporate Open Source Development Model*. ICSE'06, Shanghai, China. Maio, 20-28, 2006.

GUTWIN C., PENNER R, SCHNEIDER K. *Group awareness in distributed software development*. Em: Anais da ACM Conference on Computer Supported Cooperative Work (CSCW'04), 2004

HALL, T.; WILSON, D.; RAINER, A; JAGIELSKA, D. *Communication: The Neglected Technical Skill?* Em: *ACM SIGMIS CPR 2007 conference on Computer personnel research: The global information technology workforce, St. Louis, Missouri, USA*, 2007

HARJUMAA, L.; TERNOVEN, I.; VUORIO, P. *Improving Software Inspection Process with Patterns*. Em: *Fourth International Conference on Quality Software* - QSIC'04, 2004.

HARTER, D. E., KRISHNAN, M. S.; SLAUGHTER, S. A. *Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development*. Manage. Sci. 46, 4 (Abril, 2000), 451-466. DOI= http://dx.doi.org/10.1287/mnsc.46.4.451.12056.

HENNINGER, S. *Software Process as a Mean to Support Learning Software Organizations*. Twenty-fifth Annual NASA Software Engineering Workshop, 2000.

HENNINGER, S. *Using Software Process to Support Learning Software Organizations. 1st International Workshop on Learning Software Organizations* (LSO 1999), 1999.

HÖHN, E. N.; MALDONADO, J. C.; MENDONÇA, M.; FABRI, S. C. P. F.; BOAS, A. L. V.; TAMBASCIA, C.; FREITAS, M. E.; PAGLIUSO, P. Pbr: Transferência de tecnologia baseada em pacotes de experimentação. In: Anais do III Simpósio Brasileiro de Qualidade de Software, Brasília, 2004, p. 161–175.

HUMPHREY, W. M*anaging the software process.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

HUO, Q.; ZHU, H.; Greenwood, S. *A multi-agent software engineering environment for testing Web-based applications*. IEEE, 2003

HUPFER, S.; ROSS, S.; PATTERSON, J. *Introducing collaboration into an application development environment. Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 2004.

IBM. *Rational Unified Process*. Industrial Business Machines (IBM). Disponível em http://www-306.ibm.com/software/awdtools/rup/ . Acesso em 09/01/2010.

IEEE Standard 1012-2004. IEEE *Standard for Software Verification and Validation*. Standard 1012-2004, 2004.

IEEE Standard 610.12. IEEE *Standard Glossary of Software Engineering Terminology*. IEEE-STD-610.12, 1990.

International Organization For Standardization/ International Eletrotechnical Commission. ISO/IEC 12207 *Systems and software engineering– Software life cycle processes*, ISO, 1995.

International Organization For Standardization/ International Eletrotechnical Commission. ISO/IEC 12207 *Systems and software engineering– Software life cycle processes*, Geneve: ISO, 2008.

International Organization For Standardization/ International Eletrotechnical Comission. ISO/IEC 15504-2. *Information technology - Software process assessment* - Part 2: A reference model for processes and process capability. 2003.

International Organization For Standardization/ International Eletrotechnical Comission. *ISO/IEC 15504-5: Information Technology - Process Assessment - Part 5: An exemplar Process Assessment Model*, Geneve: ISO, 2006.

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

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *The Unified Software Development Process*. Addison Wesley, 1999.

JACCHERI, M. CONRADI, R. Techniques for Process Model Evolution in EPOS. Em: IEEE Transactions on Software Engineering, v. 19, n. 12, p. 1145-1156, 1993.

KIEL, L. *Experiences in Distributed Development: A Case Study.* Em: *International Workshop on Global Software Development at ICSE*, 2003, Oregon. Proceedings. EUA, 2003.

LARMAN, C. *Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design (3rd Edition)*. Prentice-Hall, 2004.

LARMAN, C.; BASILI, V.R. *Iterative and incremental developments a brief history.* IEEE, v. 36, n. 6 p. 47- 56, 2003.

LAPORTI, V. BORGES, M. BRAGANHOLO, V. *Athena: A collaborative approach to requirements elicitation*. Em: *Collaborative Engineering: from Concurrent Engineering to Enterprise Collaboration*, *Computers in Industry*, v. 60, n. 6, p. 367-380, 2009, ISSN 0166-3615, DOI: 10.1016/j.compind.2009.02.011.

LIN, J., FOX, M. S.; BILGIC, T.: *Requirement Ontology for Engineering Design. Enterprise Integration Laboratory*. University of Toronto, Manuscript, September, 1996.

LINDVALL, M.; MUTHIG, D.; DAGNINO, A.; WALLIN, C.; STUPPERICH, M.; KIEFER, D.; MAY, J.; KAHKONEN, T. *Agile Software Development in Large Organizations*. IEEE Computer v. 37, n. 12, p. 26-34, 2004.

LINDVALL, M.; RUS, I. *Knowledge Management for Software Organizations. Managing Software Engineering Knowledge*, Aurum, A., et al.,(eds.), Springer, Berlin, p. 73–94, 2003.

LINDVALL, M.; RUS, I.; SINHA, S. *Technology Support for Knowledge Management.* Em: *Fourth Intˇ Sl Workshop Learning Software Organizations* (LSOˇ S02), agosto, 2002.

LOPES, A.; PINHO, R. PAULOVICH, F.; MINGHIM, R. *Visual text mining using association rules*. Computers & Graphics, v. 31, n. 3, p. 316-326, 2007. DOI:10.1016/j.cag.2007.01.023

MACK, R.; RAVIN, Y.; BYRD, R. J. *Knowledge portals and the emerging digital knowledge workplace*. IBM System Journal, p. 925–955, 2001.

MAFRA, S. TRAVASSOS, G. Estudos Primários e Secundários apoiando a busca por evidência em Engenharia de Software. Relatório Técnico – ES 687/06.Programa de Engenharia de Sistemas e Computação COPPE/UFRJ. Março, 2006.

MAIDANTCHIK, C., ROCHA, A.R., XEXÉO, G., *Software Process Standardization for Distributed Working Groups*. *Fourth IEEE International Symposium and Forum on Software Engineering Standards*. Curitiba, Brasil. Maio, 1999.

MAIDANTCHIK, C. Gerência de Processos de Software para Equipes Geograficamente Dispersas. Tese de Doutoramento, COPPE/ UFRJ, Rio de Janeiro, Junho de 1999.

MALDONADO, J. C.; SANCHES, R.; FABBRI, S. Qualidade de software: Teoria e prática, v. 1, cap. Parte 1: Processo de Software (Sessões: 3.4 Verificação e validação de software; 3.5 Teste de software). 1 ed Makron Books, p. 66–84, 2001.

MALHEIROS, V. Gestão de Conhecimento e Tecnologia da Informação: A Experiência do SERPRO. Em: Anais da III Jornada Ibero-americana de Engenharia de Software e Engenharia de Conhecimento - JIISIC, 2003.

MALHEIROS, V.; PINHO, R. Uma Visão Tecnológica da Gestão de Conhecimento - Estudo de Caso no SERPRO. Em: ISKM - *International Symposium of Knowledge Management*, 2003.

MALHEIROS, V.; PAIM, F.; GUZZO, H.; MENDONÇA, M. Uma Abordagem para Melhoria Contínua do Processo de Desenvolvimento de Software. Em: Simpósio Brasileiro de Qualidade de Software 2006 (SBQS), Vila Velha - ES, 2006.

MALHEIROS, V.; HÖHN, E.; PINHO, R.; MENDONÇA, M.; MALDONADO, J. C. *A Visual Text Mining approach for Systematic Reviews*. Em: *Proceedings of International Symposium on Empirical Software Engineering and Measurement* (ESEM), Madrid, 2007a.

MALHEIROS, V.; CUNHA, L.; REHEM, S. Mantis-PMP: uma ferramenta livre para gestão de mudanças em processos. Em: Congresso SERPRO de Tecnologia e Gestão Aplicadas a Serviços Públicos - ConSerpro 2007, Porto Alegre, 2007b.

MALHEIROS, V.; PAIM, F. R.; MENDONÇA, M. *Continuous Process Improvement at a Large Software Organization*. *Software Process Improvement and Practice Journal*, v. 13, p. 1-16, 2008a.

MALHEIROS, V; REHEM, S; MALDONADO, J. C. Atabaque: uma contribuição de sucesso na evolução de processos. Em: Anais do VII Simpósio Brasileiro de Qualidade de Software (SBQS), Florianópolis, 2008b.

MALHEIROS, V.; HÖHN,E.; MALDONADO, J.C. *Qualipso Project: Quality Recommendations for FLOSS development processes: A perspective based on trustworthy elements.* Relatório Técnico RT-335, Universidade São Paulo, 2009a.

MALHEIROS, V. MALDONADO, J.C. *Qualipso Project: Open Process for Evolution, Maintenance and Administration of OMM*. Relatório Técnico RT-347, Universidade São Paulo, 2009b.

MALHEIROS, V. SEAMAN, C. MALDONADO, J. *An Approach for Collaborative and Distributed Software Process Improvement (SPI).* III Workshop de Desenvolvimento Distribuído de Software (WDDS), 2009c.

MALHEIROS, V. HÖHN, E. MESSIAS, R., MALDONADO, J.C., PETRINJA, E., SILLITI, A., WITTMAN, M. ORTEGA, F. RUFFATI, G. *WD6.5.4 - Survey Results Version 1*. Relatório Técnico Projeto Qualipso. Janeiro, 2010a.

MALHEIROS, V.; PAIM, F.; MENDONÇA, M.; MALDONADO, J.C. *Software Process in a Large Organization: Can we get agile?* Submetido ao JBCS – *Journal of the Brazilian Computer Society*, 2010b.

MALHEIROS, V., AGRA, R., ANDRADE, A. Processo Demoiselle: um processo livre para desenvolvimento de software para e-Gov. Submetido ao Workshop de Software Livre (WSL) 2010, Porto Alegre, Brasil, 2010c.

MARKKULA, M. *Knowledge Management in Software Engineering Projects. Proceedings of the International conference on Software Engineering and Knowledge Engineering*, SEKE, v. 99, p. 20–27, 1999.

MARWICK, D. *Knowledge management technology*. IBM System Journal, v. 40, n. 4, 2001.

MELIAN, C.; AMMIRATI, C.; GARG, P.; SEVÓN, G. *Building Networks of Software Communities in a Large Corporation*, 2002. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.2987

MENDONÇA, M.; MALDONADO, J.; OLIVEIRA, M.; CARVER, J.; FABBRI, S.; SHULL, F.; TRAVASSOS, G.; HÖHN, E.; BASILI, V. *A Framework for Software Engineering Experimental Replications*. Em: *13th IEEE International Conference on Engineering of Complex Computer Systems*, 2008.

MICHLMAYR, M. *Software Process Maturity and the Success of Free Software Projects*. Em: *Conference on Software Engineering: Evolution and Emerging Technologies* 2005, v. 130, p. 3-14, 2005

MOE, N., DYBA, T. *The use of an electronic process guide in a medium-sized software development company. Software Process Improvement and Practice*, v. 11, n. 1, p. 21-34. John Wiley & Sons, Ltd., 2006. Disponível em: http://dx.doi.org/10.1002/spip.250.

MONTALBANO, E. *IBM donates rational processes to eclipse. 2005*. Disponível em http://www.javaworld.com/javaworld/jw-10-2005/jw-1017-idgns-eclipse.html. Acesso em 01.02.2007

MONTANGERO, C.; DERNIAME, J.-C.; KABA, B. A.; WARBOYS, B. *The software process: Modelling and technology.* In: Software Process: Principles, Methodology, Technology, London, UK: Springer-Verlag, p. 1–14, 1999.

MONTONI, M. ROCHA, A.R. *A Methodology for Identifying Critical Success Factors that Influence Software Process Improvement Initiatives: An Application in the Brazilian Software Industry*. EuroSPI 2007, LNCS 4764, p. 175-186, 2007.

MONTONI, M.; CERDEIRAL, C.; ZANETTI, D.; ROCHA, A.R. Uma Abordagem para Condução de Iniciativas de Melhoria de Processos de Software, 2008. Disponível em: http://www.softex.br/mpsbr/_artigos/artigo.asp?id=1826. Acesso em 25.03.2010

MONTONI, M.; ROCHA, A.R.; WEBER, K. *MPS.BR: A Successful Program for Software Process Improvement in Brazil*. EuroSPI 2008 (*European Systems & Software Process Improvement and Innovation*), Dublin City University , Ireland, 2008.

NAKAGAWA, E. *An investigation of the open source development process.* Anais da IV *Jornada IberoAmericana de Inginiería del Software e Ingeniaría del Conocimento* - JIISIC, 2004.

NAKAGAWA, E. Uma contribuição ao projeto arquitetural de ambientes de engenharia de software. Tese de Doutoramento, Instituto de Ciências Matemáticas e de Computação – ICMC-USP, São Carlos, SP, 2006.

NASCIMENTO, L. MALHEIROS, V. Estratégias de Sucesso na Institucionalização de um Processo Padrão e Evolução da Maturidade do Desenvolvimento. ConSerpro: Congresso SERPRO de Tecnologia e Gestão Aplicadas a Serviços Públicos, Brasília, 2004.

NASCIMENTO, L. Uma Abordagem para Melhoria da Infraestrutura de Medições. Dissertação de Mestrado, Universidad Salvador - UNIFACS, 2006.

NIAZI, M., WILSON, D., ZOWGHI, D. *A maturity model for the implementation of software process improvement: an empirical study*. *The Journal of System and Software*, v. 74, p. 155-172, 2005a.

NIAZI, M., WILSON, D., and ZOWGHI, D. *A framework for assisting the design of effective software process improvement implementation strategies*. *The Journal of System and Software*, v.2, p. 204-222. DOI= http://dx.doi.org/10.1016/j.jss.2004.09.001, 2005b.

NIAZI, M.: Software Process Improvement: A Road to Success. In: Münch, J., Vierimaa, M. (eds.) PROFES 2006. LNCS, vol. 4034, pp. 395–401. Springer, Heidelberg (2006)

NASIR, M.H.N.M.; AHMAD, R.; HASSAN, N.H. *Resistance Factors in the Implementation of Software Process Improvement Project in Malaysia*. Journal of Computer Ciência, v.4, n. 3, p.221-219, 2008.

NONAKA, I.; TAKEUCHI, H. Criação de Conhecimento na Empresa. Ed. Campus, 1997.

NORTHROP, L. *Software product lines: Reuse that makes business sense*. Em: XX Simpósio Brasileiro de Engenharia de Software, 2006.

NUNES, V.; SOARES, A.; FALBO, R. Apoio à Documentação em um Ambiente de Desenvolvimento de Software. *VII Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software* - IDEAS'2004, 2004.

OGASAWARA, H. ISHIKAWA, T. MORIYA, T. *Practical Approach to Development of SPI Activities in a Large Organization ~Toshiba's SPI History since 2000~*. ICSE'06, 2006.

O'LEARY, D. *Enterprise knowledge management*. IEEE Computer, p. 54–61, 1998a.

O'LEARY, D. *Knowledge management systems: Converting and connecting*. IEEE Intelligent Systems, v. 13, n. 3, p. 30–33, 1998b.

O'LEARY, D. *Using AI in Knowledge Management: Knowledge Bases and Ontologies*. IEEE *Intelligent Systems*, v. 13, n. 3, p. 34–39, 1998c.

OLIVEIRA, K. Modelo para Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio. Tese de Doutoramento, COPPE–UFRJ, 1999.

OMG. *Software & Systems Process Engineering Meta-Model Specification, Version 2.0. Object Management Group (OMG).* Abril, 2008. Disponível em: http://www.omg.org/technology/documents/formal/spem.htm. Acesso em 05.08.2009.

OMG. *Business Process Model and Notation (BPMN), version 1.2. Object Management Group (OMG),* Janeiro, 2009. Disponível em: http://www.omg.org/spec/BPMN/1.2/PDF/

OSTERWEIL, L. *Software processes are software too (revised)*. Em: *Proceedings of the International Conference on Software Engineering*, 1997.

PASSOS, M. C. Uma Ferramenta para o Mapeamento do Conhecimento e suas aplicações. Dissertação de Mestrado, Mestrado em Redes de Computadores, Universidade Salvador, Salvador, 2004.

PAULK, M.; WEBER, C.; GARCIA, S.; CHRISSIS, M.; BUSH, M. *Key Practices of the Capability Maturity Model SM*, *Version 1.1*, 1993. Disponível em www.sei.cmu.edu/cmm/ Acesso em 30.06.2006.

PAULK, M. *Agile Methodologies and Process Disciplines*. The Journal of Defense Software Engineering. 2002. Disponível em: http://www.stsc.hill.af.mil/crosstalk/2002/10/paulk.html

PFLEEGER, S. *Understanding and improving technology transfer in software engineering*. The Journal of Systems & Software, v. 47, n. 2-3, p. 111–124, 1999.

PINHO, R. ; OLIVEIRA, M. C. ; LOPES, A. *An incremental space to visualize dynamic data sets. Multimedia Tools and Applications*, Springer Netherlands, 2010 doi: 10.1007/s11042-010-0483-5 http://www.springerlink.com/content/vh5h56303173k366/

PRESSMAN, R. Engenharia de Software. Editora McGraw-Hill, ISBN: 8586804576, 2006.

PRIKLADNICKI, R. MuNDDoS: Um Modelo de Referência para Desenvolvimento Distribuído de Software. Dissertação de Mestrado, PPGCC-PUCRS, 2003.

PRIKLADNICKI, R.; AUDY, J.; EVARISTO, R. *Global Software Development in Practice: Lessons Learned*. *Software Process Improvement and Practice*, v. 8, n. 4, p. 267-281, 2004.

PRIKLADNICKI, R.; AUDY, J. L. N. Uma análise comparativa de práticas de desenvolvimento distribuído de software no Brasil e no exterior. Em: Anais do XX Simpósio Brasileiro de Engenharia de Software (SBES), p. 255–270, 2006.

PROJECT MANAGEMENT INSTITUTE (PMI) *A Guide to the Project Management Body of Knowledge*. Pensylvania, US: Project Management Institute, 2004.

PROTÉGÉ. *Welcome to Protégé*. 2007. Disponível em http://protege.stanford.edu/ (Acessado em 2007.01.10)

QUALIPSO Project. *Deliverable A6.D2.6.2 - Trustworthy elements identified in OS processes*. Relatório Técnico. Outubro, 2008.

QUALIPSO Project. *Quality platform for open source software*. Relatório Técnico, QUALIPSO Project, 2005.

RAINER, A.; HALL, T. *Key success factors for implementing software process improvement: a maturity-based analysis*. *Journal of Systems and Software*, 2002.

RAMASUBBU, N. e BALAN, R. *Globally Distributed Software Development Project Performance: An Empirical Analysis*. ESEC-FSE'07, Croatia, 2007

RAYMOND, E. *The Cathedral and the Bazaar. Knowledge, Technology, and Policy*. v. 12, n. 3, p. 23–49, 1999.

REIS, Carla. Uma Abordagem Flexível para Execução de Processos de Software Evolutivos. Tese de doutoramento. Orientador: Daltro Nunes. UFRGS, 2003a.

REIS, Christian. Caracterização de um Processo de Software para Projetos de Software Livre. Dissertação de Mestrado, ICMC, Universidade São Paulo, 2003b. Disponível em http://async.com.br/~kiko/dissert_usp.pdf. Acesso em 06.04.2006.

RIEHLE, D., ELLENBERGER, J., MENAHEM, T., MIKHAILLOVSKI, B., NATCHETOI, Y., NAVEH, B., ODENWALD, T. *Open Collaboration within Corporations Using Software Forges*. IEEE Software, 26, 2, 52-58, 2009. DOI= http://dx.doi.org/10.1109/MS.2009.44

RISING, L. *Patterns: A way to reuse expertise*. 2004 Disponível em http://www.agcs.com/supportv2/techpapers/patterns/papers/*expertise*.htm. Acesso em 01.10.2005

ROBERTS, D.; JOHNSON, R. *Evolving Frameworks. A Pattern Language for Developing. Object-Oriented Frameworks*, 2005. Disponível em http://st-www.cs.uiuc.edu/users/ droberts/evolve.html. Acesso em 01.09.2005

RUS, I.; LINDVALL, M. *Knowledge management in software engineering*. Software, IEEE, v. 19, n. 3, p. 26–38, 2002.

RUS, I.; LINDVALL, M.; SINHA, S. *Knowledge Management in Software Engineering: A State of the Art Report. Data & Analysis Center for Software: ITT Industries, Rome*, NY, 2001.

SALVIANO, C. F. Introdução à melhoria de processo de software com ISO/IEC 15504 e CMMI. Relatório Técnico, CenPRA - TRT1351, 2004.

SANTOS, A. R. Gestão do Conhecimento: Uma experiência para o sucesso Empresarial. Editora Champagnat, 2001.

SANTOS, G., MONTONI, M., FIGUEIREDO, S., ROCHA, A.R. SPI-KM - *Lessons Learned from Applying a Software Process Improvement Strategy Supported by Knowledge Management*. Lecture Notes in Computer Science, v. 4589/2007, p. 81-95, 2007.

SCAMPI Upgrade Team. *Standard CMMI Appraisal Method for Process Improvement* (SCAMPI), *Version 1.2*: *Method Definition Document*, 2006. Acesso em 01.03.2010 Disponível em http://www.sei.cmu.edu/publications/documents/01.reports/01hb001.html.

SCHWABER, K. *SCRUM Development Process*, 2006. Disponível em: http://jeffsutherland.com/oopsla/schwapub.pdf. Acesso em 06/01/2010.

SEI - Software Engineering Institute. *Capability Maturity Model Integration for Development (CMMI-DEV)*, *Version 1.2*. *Technical Report CMU/SEI-2006-TR-008. Software Engineering Institute (SEI)*, Carnegie Mellon University, 2006. Disponível em: http://www.sei.cmu.edu/reports/06tr008.pdf (Acesso em: 14/11/2009)

SERPRO.Gestão do Conhecimento, 2003. Disponível em: http://www1.SERPRO.gov.br/publicacoes/gco_site/m_sumario.htm. Acesso em: 27.03.2010.

SERPRO. Sistemática SERPRO de Ensino à Distância. Procedimentos Internos de Educação à Distância. SERPRO, 2000. Disponível em http://www.SERPRO.gov.br. Acesso em 14/11/2009.

SERPRO. PSDS - Processo SERPRO de Desenvolvimento de Soluções. Acesso restrito ao SERPRO, 2007.

SERPRO. Sistemática das Avaliações de Maturidade e SERPRO (versão 3.0). Acesso restrito ao SERPRO, 2006.

SHNEIDERMAN, B. PLAISANT, C. *Designing the User Interface: Strategies for Effective Human-Computer Interaction, 4th Edition*. Addison-Wesley, 2004.

SHULL, F.; CARVER, J.; TRAVASSOS, G. H. *An empirical methodology for introducing software processes*. Em: *Proceedings of the 8th European Software Engineering Conference, 9th ACM SIGSOFT international Symposium on Foundations of Software Engineering* (ESEC/FSE-9), ACM. Austria, 2001. DOI= http://doi.acm.org/10.1145/503209.503248.

SHULL, F.; MENDONÇA, M.; BASILI, V.; CARVER, J.; MALDONADO, J. C.; FABBRI, S.; Travassos, G.; FERREIRA, M. C. *Knowledge-sharing issues in experimental software engineering. Empirical Software Engineering,* v. 9, p. 111–137, 2004.

SILVA, B.; FALBO, R. Definição de um processo padrão para software livre. In: Anais do V Simpósio Brasileiro de Qualidade de Software (SBQS 2006)., Vila Velha, ES, 2006.

SOFTEX. O Impacto do Software Livre e de Código Aberto na Indústria de Software do Brasil. Softex, 2005.

SOFTEX. MPS.BR - Guia de Avaliação. 2006. Disponível em http://www.softex.br/mpsbr/_guias/MPS.BR_GUIA_DE_AVALIACAO_V1.0.pdf (Acessado em 2007.02.03)

SOFTEX. MPS.BR - Guia Geral. 2009. Acesso em outubro, 2009. Disponível em: http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_2009.pdf.

SOFTEX. Estrutura Organizacional. Disponível em: http://www.softex.br/mpsBr/_outros/_estrutura.asp. Acesso em: 20/03/2010.

SOMMERVILLE, I. SAWYER, P., VILLER, S. *Viewpoints for requirements elicitation: a practical approach*. Em: *Third IEEE International Conference on Requirements Engineering* (ICRE 98), 1998.

SOMMERVILLE, I. *Software Engineering Sixth*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2001.

STELZER, D. MELLIS, W. *Success Factors of Organizational Change in Software Process Improvement*. *Software Process Improvement and Practice*, Volume 4, Issue 4 John Wiley & Sons, Ltd., 1999.

STOREY M., CUBRANIC, D., GERMAN D. *On the use of visualization to support awareness of human activities in software development: a survey and a framework*. Em: Anais da ACM Symposium on SoftwareVisualization (SoftVis'05), 2005.

STEIN, M., RIEDL, J., HARNER, S. J., MASHAYESKHI, V. *A case study of distributed, asynchronous software inspection.* Em: Anais da *19th international Conference on Software Engineering*. ICSE '97, 1997. DOI= http://doi.acm.org/10.1145/253228.253250

THIRY, M. von WANGENHEIM, C.; ZOUCAS, A. PICKLER, K. Uma Abordagem para a Modelagem Colaborativa de Processos de Software em Micro e Pequenas Empresas. SBQS, 2006.

TRAVASSOS, G.; GUROV, D. AMARAL, E. Introdução à Engenharia de Software Experimental. Relatório Técnico do Programa de Engenharia de Sistemas e Computação (COPPE- UFRJ), 2002.

TRAVASSOS, G.; KALINOWSKI, M. iMPS: Resultados de desempenho de organizações que adotaram o Modelo MPS. Campinas, SP : Associação para Promoção da Excelência do Software Brasileiro – SOFTEX, 2008.

VILLELA, K. Definição e construção de ambientes de desenvolvimento de software orientados a organização. Tese de Doutoramento, Universidade Federal do Rio de Janeiro UFRJ, 2004.

WANG, D.; KING, D. *Software Engineering Processes: Principles and Applications*. CRC Press, 2000. ISBN: 978-0-8493-2366-9.

WARD, J.; AURUM, A. *Knowledge management in software engineering – describing the process. Proceedings of Software Engineering Conference*, p. 137–146, 2004.

WERNER, C., MANGAN, M., MURTA, L., PINHEIRO, R., MATTOSO, M., BRAGA, R. BORGES, M. *OdysseyShare: An environment for collaborative component-based development*. Em: *IEEE Conference on Information Reuse and Integration (IRI)*, 2003.

WIEGERS, K. E. *Read my lips: new models*. IEEE Software, v. 15, n. 5, p. 10–13, 1998.

WIKIPEDIA. *Wiki*. 2010a. Disponível em http://en.wikipedia.org/wiki/Wiki. Acesso em 25.03.2010)

WIKIPEDIA. *Computer supported cooperative work*. 2010b Disponível em: http://en.wikipedia.org/wiki/Computer_supported_cooperative_work. Acesso em: 27/3/2010

WHITEHEAD, J. *Collaboration in Software Engineering: A Roadmap*. Em: *Future of Software Engineering* 2007. *International Conference on Software Engineering*. IEEE Computer Society, Washington, DC, 214-225, 2007. DOI= http://dx.doi.org/10.1109/FOSE.2007.4

WITTMANN, M., NAMBAKAM, R., PETRINJA, E., ORTEGA, F., RUFFATI, G., OLTOLINA, S., MALHEIROS, V. *Working Document WD6.3.1 - CMM-like model for OSS, version 1*. Relatório Técnico Projeto Qualipso. Dezembro, 2009. Disponível em: http://www.qualipso.org/sites/default/files/A6.D1.6.3CMM-LIKEMODELFOROSS.pdf

YEAKLEY C., FIEBRICH, J. *Collaborative Process Improvement: With Examples from the Software World.* Wiley-IEEE Computer Society Press, 2007.

ZANETTI, D., MONTONI, M., ROCHA, A. R. *Benchmarking* em Iniciativas de Melhorias em Processos de Software. Simpósio Brasileiro de Qualidade de Software 2009 (SBQS), 2009.

ZHAO, W.; KEARNEY, D. *Deriving architectures of Web-based applications. Lecture Notes in Computer Science.* Springer, 2003.

Apêndice A

Neste apêndice são apresentados alguns exemplos de artefatos e técnicas utilizados em um programa de MPS que servem para ilustrar como pode ser gerenciado um programa de MPS e como detalhes dos mecanismos propostos podem ser implementados. Um resumo da política de publicação do processo livre Demoiselle também é apresentado (https://sourceforge.net/projects/demoiselle-proc/). O detalhamento da política está disponível no Sourceforge.

A1. Exemplo de EAP para gerenciar a MPS

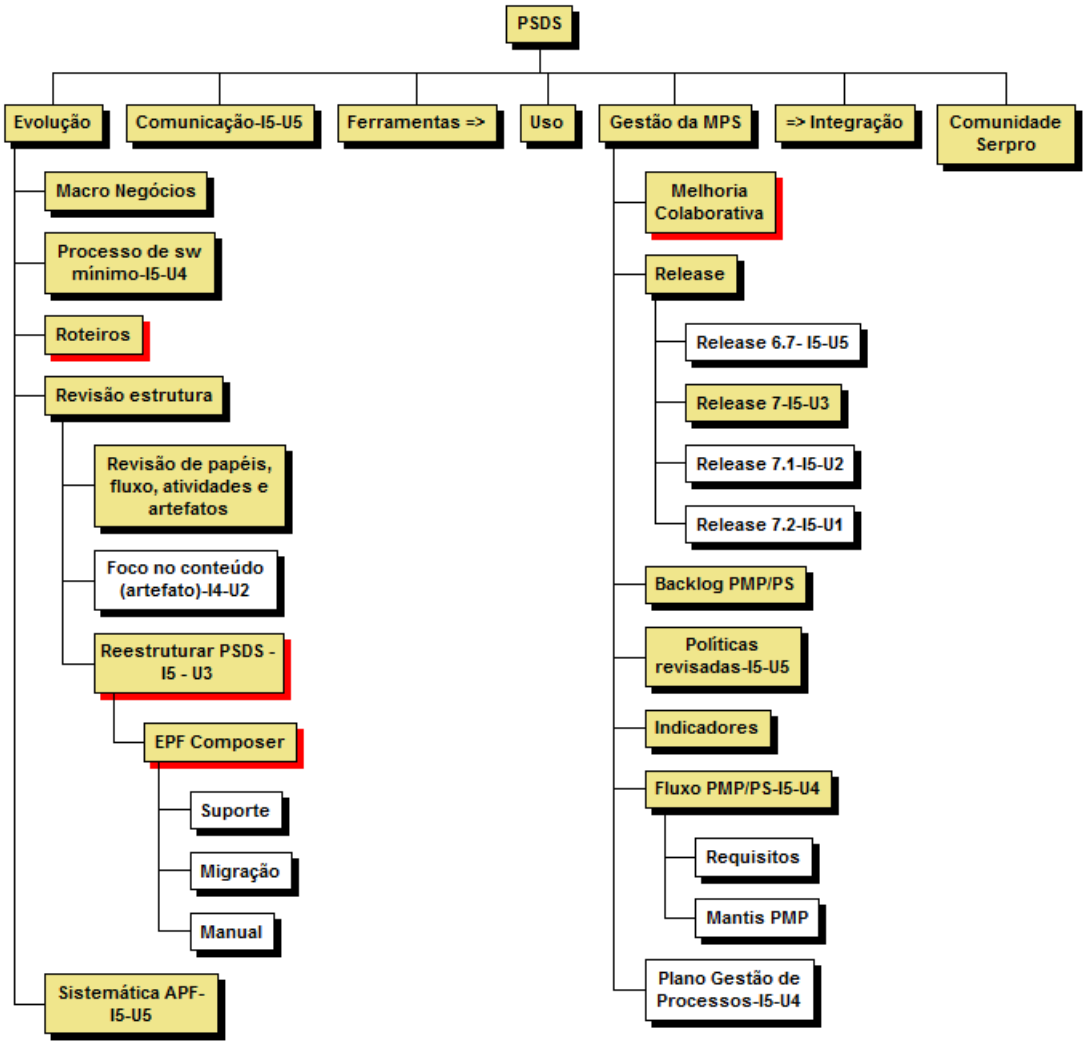


Figura 61: Exemplo de EAP utilizada no planejamento de um programa de MPS

A2. Exemplo de política de publicação de processo

Política de Publicação do Processo Demoiselle[57]

Tipos de *Releases*

Principal
Uma release principal do Processo Demoiselle contempla mudanças estruturalmente significativas em relação à release imediatamente anterior, normalmente alterando sua estrutura geral ou reduzindo seu escopo (eliminação de atividades, papéis ou disciplinas). Assim, uma release principal, normalmente, é incompatível com a release imediatamente anterior.

Secundária
Uma *release* secundária do Processo Demoiselle contempla mudanças estruturalmente pouco significativas e que não geram incompatibilidade com a versão imediatamente anterior, como correções de erros e ampliação de escopo (adição de atividades, orientações, papéis ou disciplinas).

Status de *Releases*

Beta
Uma *release* do Processo Demoiselle em status *beta* é uma versão cuja *release* principal está sendo disponibilizada para testes, ou seja, aguardando validação ou sendo validada em projetos pilotos.

Estável
Uma *release* do Processo Demoiselle em status *estável* é uma versão cuja *release* principal já foi testada e validada em pelo menos um projeto piloto. Assim, toda *release* principal estável é precedida por uma versão beta correspondente. Além disso, uma versão estável nunca voltará a ter status de beta.

Número de Versão de *Releases*
O número de versão de uma *release* do Processo Demoiselle possui o formato XX.YY[b|e], onde *XX* representa o número de versão da *release* principal, *YY* representa o número de versão da *release* secundária e *[b|e]* indica se a versão da *release* é beta ou estável. Assim, a *release* com número de versão *1.0b* representa a versão beta da *release* principal de número *1* e *release* secundária de número *0*, assim como a *release* com número de versão *2.14e* representa a versão *estável* da *release* principal de número *2* e *release* secundária de número *14*.
Não deve haver duas *releases* com as mesmas versões principal e secundária e com status diferentes, como, por exemplo, uma *release 1.0b* e outra *1.0e*. Assim, mesmo que não tenha havido mudanças na versão *1.0b*, mas apenas uma validação da mesma em projetos pilotos, a versão estável desta deve ser numerada como *1.1e*.
Note também que, como o status de beta está relacionado apenas à versão da *release* principal, uma *release* secundária beta não pode suceder uma *release* secundária estável. Por exemplo, a versão *1.4b* não pode suceder a versão *1.3e*, porque a versão da *release* principal *1* já estava estável na versão *1.3e* e não pode voltar a ser beta na versão *1.4b*. Assim, se uma *release* que seria secundária necessitar de testes em piloto, ela deve ser revista e considerada como uma *release* principal. Nesse caso, a nova *release* seria 2.0b para o exemplo dado.

Plano de Suporte a *Releases*
Cada *release* principal estável terá suporte durante todo o período desde a sua disponibilização até o fim de 1 ano contado a partir da data de disponibilização da *release* principal estável subsequente. Além disso, o suporte a uma *release* principal se estende a todas as suas respectivas *releases* secundárias. Assim, se a *release* atual é a 1.4e e a próxima *release* principal (versão 2.YYe) será disponibilizada em 01 de janeiro de 2050, todas as *releases* 1.YYe disponibilizadas até 31 de dezembro de 2049 terão suporte até o dia 31 de dezembro de 2050, ou seja, exatamente 1 ano após a disponibilização da primeira versão estável da próxima *release* principal.

[57] A autora desta tese é co-autora dessa política, cuja primeira versão foi escrita por empregados do SERPRO.

Fases, Atividades, Documentos e Papéis envolvidos numa disponibilização de Release

A disponibilização de uma *release* envolve vários papéis desempenhando atividades de verificação, correção, atualização e criação de documentos. Entre esses documentos estão a própria política de publicação, a política de contribuição e as notas de *release*. Além disso, outras atividades como definir datas e projetos pilotos, comunicar usuários através de e-mail e do sítio Web, acompanhar o teste de uma versão beta e analisar riscos de migração para a nova versão também fazem parte do ciclo de disponibilização de uma *release*.

A Figura 1 apresenta o fluxo desde o planejamento até a publicação e teste de uma *release* do Processo Demoiselle. Ele é composto por cinco fases: planejamento, desenvolvimento, validação, publicação e piloto.

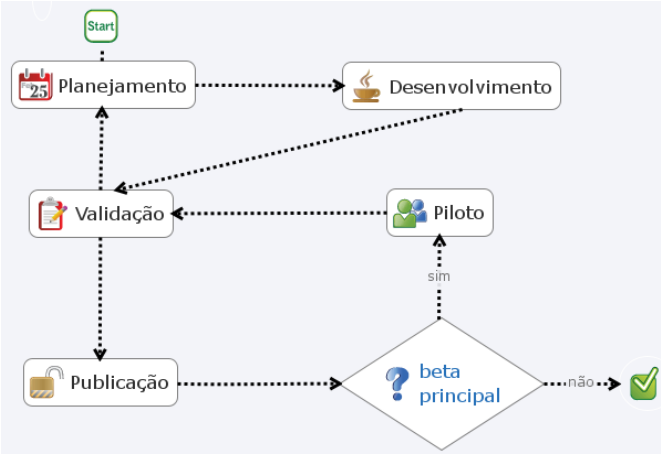


Figura 1: Fluxo de desenvolvimento e publicação de uma release do Processo Demoiselle

A3. Exemplos de notificações para o tratamento PMPs

Tabela 13: Notificações automáticas para o tratamento de PMPs[58]

Ação	Estado Inicial	Estado Final	Mensagem	Notificados
Submeter PMP	Em Registro	Em Pré-Análise	PMP xxx submetida para análise do SEPG Local	(2)
Submeter PMP	Em Registro	Em Análise	PMP xxx submetida para análise do grupo especialista ou SEPG Corporativo	(3) (4)
Rejeitar PMP	Em Pré-Análise	Rejeitada	PMP xxx rejeitada	(1)
Registrar parecer	Em Pré-Análise	Em Pré-Análise	Parecer registrado pelo GPES Local	(1) (2)
Rever rejeição	Rejeitada	Em Pré-Análise	PMP xxx submetida novamente para análise do GPES Local	(1)
Registrar parecer	Em Análise	Em Análise	Parecer registrado pelo GE/Corporativo	(1) (2) (3) (4)
Rever rejeição	Rejeitada	Em Análise	PMP xxx submetida novamente para análise do GE/Corporativo	(1) (2) (3) (4)
Registrar colaboração/ solicitação	Qualquer estado (a exceção de em registro)	Qualquer estado (a exceção de em registro)	Mensagem redigida pelo usuário	Endereçados escolhidos pelo usuário
Aprovar PMP	Em Análise	Aprovada	PMP xxx aprovada	(1) (2) (3) (4)
Aprovar para a release	Implementada	Aprovada para release	PMP xxx será disponibilizada na release <número_da_release>	(1) (2) (3) (4)
Cancelar PMP	Qualquer estado	Cancelada	PMP xxx cancelada pelo motive <motivo>	(1) (2) (3) (4)

(1) Proponente; (2) Coordenador SEPG Local (SEPG na unidade); (3) Coordenador do grupo especialista; (4) SEPG corporativo

[58] Essa lista de notificações foi elaborada por Henrique Guzzo, e apenas adaptada pela autora desta tese.

A4. Exemplo de Plano para a MPS

Histórico de Versões

Data	Versão	Descrição	Autor	Revisor	Aprovado por
	1.0	Elaboração da versão inicial do plano	Maria	Jorge, Tadeu, Marta, Pedro	
	1.1	Atualização do plano com as sugestões da equipe e com o retorno da revisão por par	Maria	Pedro	
	1.2	Atualização após revisão de GQS	Maria	Marta	
	1.3	Atualização em revisão com o patrocinador	Maria	Marta	Joana

ÍNDICE

1. Identificação
2. Objetivos e metas estratégicas da melhoria
3. Estrutura para a Melhoria do Processo
4. Papéis e Responsabilidades
5. Abordagem Estratégica
6. Cronograma da Melhoria
7. Sensibilização
8. Medições
9. Plano de Comunicação
10. Ferramentas envolvidas
11. Plano de Releases
12. Plano de Análise de Decisão e Resolução
13. Plano de Avaliações
14. Identificação de Riscos
15. Mecanismo de Acompanhamento deste Plano
16. Observações

Plano de Gestão do Processo Corporativo

1. Identificação

Coordenação PSDS	
GPES Corporativo:	
GQS Corporativo:	
Responsável :	
Patrocinador :	
Período :	

2. Objetivos e metas estratégicas da melhoria

São diretrizes dos objetivos e metas...

São vetores para a evolução do processo...

São objetivos estratégicos da melhoria...

1. Estrutura para a Melhoria do Processo

1.1. Organograma da Empresa

1.2. Estrutura de Apoio ao processo

2. Papéis e Responsabilidades

	Papel envolvido	Unidade	Responsável
1	**Grupos Especialistas**		
	Responsabilidades no processo		

2	**Coordenação corporativa (GPES, GQS)**		
	Responsabilidades no processo		

3	**GPES Locais**		
	Responsabilidades no processo		

4	**Coordenações de desenvolvimento**		
	Responsabilidades no processo		

3. Abordagem Estratégica

3.1. Estratégia para Evolução do PSDS

3.2. Estratégia para a internalização do PSDS nas unidades

3.3. Estratégia para Liberação de Novos Releases do Processo

3.4. Estratégia para recepção de novos concursados

3.5. Estratégia para integração com outros projetos da empresa

4. Cronograma da Melhoria

Produto	Atividade	Marco	Recursos

5. Sensibilização

Nome do Evento	Público Alvo	Período	Coordenador

6. Medições

6.1. Indicadores

Uma descrição detalhada dos indicadores, está disponível no PMA corporativo.

6.2. Custos

Item	Quantidade	Unidade	Custo Unitário (R$)	Valor Final
Consultoria		HH	n/a	
Viagens		U.N.	n/a	
Treinamentos externos		U.N.	n/a	
Revisão independente de GQS corporativa		U.N.	n/a	
Mentoria de Institucionalização		U.N.	n/a	
Pré-Avaliação		U.N.	n/a	
Avaliação Interna de Maturidade		U.N.	n/a	
Avaliação Oficial		U.N.	n/a	
			Total	

6.3. Estimativas de Esforço e Custo para os GPES e GE

Esforço do GE (HD)	Líder	Analista	Programador	Apoio	Total
Engenharia de Produtos de Software	0		0	0	
Testes/Homologação	0		0	0	
...					
Total	0		0	0	

	Líder	Analista	Programador	Apoio
Custo Unitário				

Custos do GE	Líder	Analista	Programador	Apoio	Total
Engenharia de Produtos de Software	0		0	0	
Testes/Homologação	0		0	0	
...					
Total					

7. Plano de Comunicação

Informações a Divulgar	Forma de Divulgação	Periodicidade	Responsável	Público Alvo

8. Ferramentas Envolvidas

Ferramenta	Finalidade	Macro	Versão	Fornecedor

8. Plano de Releases

Release	Principais modificações	Lançamento

10.Plano de Análise de Decisão e Resolução

11.Plano de Avaliações

12. Identificação de Riscos

No.	Nível de Impacto	Fator de Risco	Ação	Responsável	Data

13. Mecanismo de Acompanhamento deste Plano

Objetivo do Controle	Periodicidade

14. Observações

A5. Passos para implantar a MPS

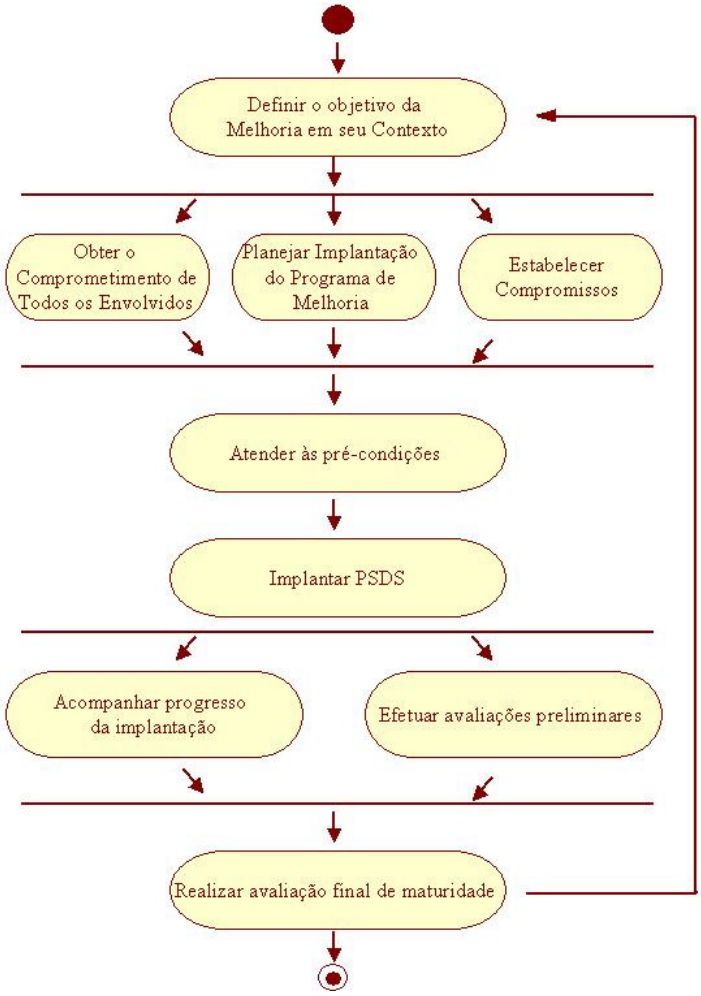


Figura 62: Estratégia para a MPS. Adaptada de Nascimento e Malheiros (2004)