UNIVERSIDADE FEDERAL DE LAVRAS

IMPLEMENTAÇÃO DO PERSONAL SOFTWARE PROCESS NO INSTITUTO CENTRO-OESTE DE DESENVOLVIMENTO DE SOFTWARE

CRISTIANO LEHRER

LAVRAS
MINAS GERAIS – BRASIL
2007

CRISTIANO LEHRER

IMPLEMENTAÇÃO DO PERSONAL SOFTWARE PROCESS NO INSTITUTO CENTRO-OESTE DE DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação *Lato Sensu* Produção de Software, com Ênfase em Software Livre, para obtenção do título de especialização.

Profa. Ângela Maria Alves, Dr.

LAVRAS

MINAS GERAIS – BRASIL

2007

CRISTIANO LEHRER

IMPLEMENTAÇÃO DO PERSONAL SOFTWARE PROCESS NO INSTITUTO CENTRO-OESTE DE DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação *Lato Sensu* Produção de Software, com Ênfase em Software Livre, para obtenção do título de especialização.

APROVADA em 16 de novembro de 2007.

Profa. Ângela Maria Alves, Dr.

LAVRAS
MINAS GERAIS – BRASIL
2007

“Vê mais longe a gaivota que voa mais alto.”

RICHARD BACH

Para Michelle.

Agradeço a todos que auxiliaram, direta
ou indiretamente, no desenvolvimento
desse trabalho, tornando-o uma
realidade ao invés de uma pretensão.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	Application Programming Interface
CenPRA	Centro de Pesquisas Renato Archer
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
FSF	Free Software Fundation
GO	Goiás
ICODES	Instituto Centro-Oeste de Desenvolvimento de Software
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
J2EE	Java 2 Plataform, Enterprise Edition
JSF	Java Server Faces
JSP	Java Server Pages
MER	Modelo Entidade-Relacionamento
MG	Minas Gerais
MIT	Massachusets Institute of Technology
MPS	Melhoria de Processo do Software
MPS.BR	Melhoria de Processo do Software Brasileiro
MR-MPS	Modelo de Referência da Melhoria de Processo do Software
MR-MPS.BR	Modelo de Referência da Melhoria de Processo do Software Brasileiro
PSP	Personal Software Process
SDK	Software Development Kit
SEI	Software Engineering Institute
SL/CA	Software Livre/Código Aberto
SOFTEX	Associação para Promoção da Excelência do Software Brasileiro
SP	São Paulo
SW-CMM	Capability Maturity Model for Software
TSP	Team Software Process
UFLA	Universidade Federal de Lavras
UML	Unified Modelling Language
USL	Unix System Laboratories
XP	Extreme Programming

Implementação do *Personal Software Process* no Instituto Centro-Oeste de Desenvolvimento de Software

Cristiano Lehrer[1,2], Ângela Maria Alves[2,3] (Orientadora)

[1]Instituto Centro-Oeste de Desenvolvimento de Software – ICODES
Rua Herculano Lobo, 206, Sala 01 – 73.801-260 – Formosa – GO – Brasil

[2]Curso de Pós-Graduação Lato Sensu Produção de Software, com Ênfase em Software Livre da Universidade Federal de Lavras (UFLA) Lavras – MG – Brasil

[3]Centro de Pesquisas Renato Archer – CenPRA
Rodovia Dom Pedro I, km 143,6 – 13.069-901 – Campinas – SP – Brasil

cristiano@icodes.org.br, alvesam@gmail.com

Resumo. *O presente artigo apresenta o desenvolvimento de uma ferramenta Web open source para os níveis de maturidade PSP 0 e PSP 0.1 do Personal Software Process (PSP), desenvolvido no Instituto Centro-Oeste de Desenvolvimento de Software (ICODES), para auxiliar a organização na obtenção do nível de maturidade Parcialmente Gerenciado (G) do modelo de referência da Melhoria de Processo do Software Brasileiro (MPS.BR).*

1. INTRODUÇÃO

Qualidade é um dos assuntos mais importantes na indústria de software da atualidade, e provavelmente será ainda mais importante num futuro próximo, pois há cada vez mais sistemas controlados por software, fazendo com que a economia de praticamente todos os países sejam muito dependentes da qualidade dos software por eles utilizados [Vasconcelos, 2003].

Entretanto, a qualidade dos produtos de software está fortemente relacionada à qualidade do processo de software. A preocupação com o processo de software está relacionada à necessidade de entender, avaliar, controlar, aprender, comunicar, melhorar, predizer e certificar o trabalho dos colaboradores da área de software [Rocha, 2001].

Muitas organizações têm alcançado melhorias significativas nos seus processos e modos de trabalho através da aplicação de um modelo de gerenciamento de processo de software, entretanto, percebem que para obterem índices melhores dependiam, ainda, do talento individual de seus colaboradores, de forma que estudos e pesquisas direcionados para a capacitação e melhoria do processo individual fossem realizados [Albuquerque, 2004].

Almejando alcançar uma melhoria no processo individual de seus colaboradores, o Instituto Centro-Oeste de Desenvolvimento de Software (ICODES) resolveu desenvolver uma ferramenta *Web open source* para auxiliar na implantação do *Personal Software Process* (PSP), níveis de maturidade PSP 0 e PSP 0.1, visando o cumprimento dos requisitos necessários para a obtenção do nível de maturidade Parcialmente Gerenciado (G) do modelo de referência da Melhoria de Processo do Software Brasileiro (MPS.BR).

O artigo está estruturado considerando uma abordagem dedutiva, começando pela exploração dos conceitos básicos que norteiam o movimento de estruturação da elaboração dos processos de software para uma abordagem mais fabril, apresentando inclusive os principais conceitos pertinentes ao *Personal Software Process* e a Melhoria de Processo do Software Brasileiro.

2. INSTITUTO CENTRO-OESTE DE DESENVOLVIMENTO DE SOFTWARE

O Instituto Centro-Oeste de Desenvolvimento de Software, nome fantasia da Sociedade para o Desenvolvimento da Tecnologia da Informação e Comunicação do Centro-Oeste, localizado na cidade de Formosa, estado de Goiás, é uma sociedade civil, de direito privado, sem fins lucrativos, que tem por missão e principal objetivo a execução de pesquisa, ensino, desenvolvimento e inovação em tecnologias da informação e comunicação, visando promover o desenvolvimento sócio-econômico e o aumento da competitividade do setor produtivo de software brasileiro, mormente da indústria de software e serviços de informática sediada no Estado de Goiás [ICODES, 2007].

Os principais objetivos do Instituto Centro-Oeste de Desenvolvimento de Software são o desenvolvimento de atividades de pesquisa e desenvolvimento, com foco nas tecnologias da informação (inclusive informática e telecomunicações), e a fomentação, promoção, apoio e execução de atividades técnicas, científicas e mercadológicas, de inovação, de geração e transferência de tecnologias e de promoção de capital humano nos temas de gestão empresarial, de *marketing* e de tecnologias de software e suas aplicações [ICODES, 2006].

O Instituto Centro-Oeste de Desenvolvimento de Software iniciou suas atividades no dia primeiro de março de 2006, e conta atualmente com a colaboração de nove colaboradores, que realizam serviços tecnológicos na área de desenvolvimento de sistemas corporativos voltados para a Internet/Intranet, utilizando preferencialmente ferramentas livres (*open source*) [ICODES, 2007].

Dentre as principais tecnologias utilizadas pelo Instituto Centro-Oeste de Desenvolvimento de Software para a construção de sistemas corporativos, destaca-se as pertencentes ao *framework* Java 2 Plataform Enterprise Edition (J2EE), como o Java 2 SDK (*Software Development Kit*) Standard Edition, Java Server Pages (JSP), Java Servlet API (*Application Programming Interface*), Enterprise JavaBeans, Hibernate, Structs, Java Server Faces (JSF) e JUnit [ICODES, 2007].

3. FÁBRICA DE SOFTWARE

Alguns anos atrás, não havia processo disciplinado para o desenvolvimento de software, que era desenvolvido de forma totalmente artesanal, sem o auxílio de técnicas, modelos e metodologias para a sua produção, ocasionando diversos problemas para a indústria de software, como por exemplo, atrasos na entrega dos projetos, altos custos de produção e baixa qualidade dos produtos desenvolvidos [Vasconcelos, 2003].

O software tornou-se o elemento-chave da evolução dos sistemas e produtos baseados em computador. No decorrer das últimas décadas, o software evoluiu de uma ferramenta de análise de informações e de resolução de problemas especializados para uma indústria em si mesma [Pressman, 1995].

Com o intuito de agilizar e melhorar a produção de software, surgiu o conceito de Fábrica de Software, que são organizações que provêm serviços de desenvolvimento de sistemas com alta qualidade, a baixo custo e de forma rápida,

através da utilização de um processo de desenvolvimento de software bem definido, e de tecnologia de ponta, além de algumas formas de retorno para reconhecer e lidar com oportunidades de melhoria do processo [Marques, 2003; Herbsleb, 1999].

Com o advento do conceito de Fábrica de Software e com os resultados positivos obtidos no desenvolvimento de software utilizando tal conceito de produção, juntamente com o esforço de redução de custos das empresas, surgiu o conceito de *outsourcing* de sistemas, que é uma operação de desenvolvimento e manutenção de software terceirizada com alguns atributos de operação fabril [Fernandes, 2004].

A terceirização de serviços de desenvolvimento de software no Brasil através da utilização de Fábricas de Software ainda é uma atividade muito recente. Contudo, empresas e pesquisadores da área de engenharia de software vêm atentando para o grande negócio que há por trás da manufatura de software em fábricas [César, 2003].

A idéia de Fábrica de Software surgiu há mais de 30 anos, sendo lapidada desde então. As primeiras fábricas foram criadas no final da década de 60, mas ainda há controversa em relação ao termo Fábrica de Software, quando o desenvolvimento de software é comparado à produção em massa de produtos industriais [Marques, 2003].

O pressuposto básico de uma Fábrica de Software é que a mesma deve possuir processos, constituídos de fases, entradas, saídas e papéis dos profissionais claramente definidos, de modo que as anomalias sejam identificadas e corrigidas previamente evitando assim a sobreposição de papéis gerencias e técnicos.

O escopo de fornecimento selecionado para a Fábrica de Software do Instituto Centro-Oeste de Desenvolvimento de Software, conforme a classificação de Fernandes [2004], é de uma fábrica de projetos de software, constituído das etapas de projeto conceitual, especificação lógica, projeto detalhado, construção e testes unitários, de integração e de aceitação, conforme apresentado na figura 01.

Figura 01 – Escopo de fornecimento da fábrica de software

Para comprovar a eficiência e a eficácia do processo de desenvolvimento de software, as Fábricas de Software buscam implementar certificações que demonstram a qualidade dos produtos desenvolvidos pela Fábrica. As certificações mais exigidas das Fábricas de Software atualmente pelo mercado, como garantia de qualidade dos seus processos de desenvolvimento, são a ISO (*International Organization for Standardization*) e a SW-CMM (*Capability Maturity Model for Software*) [Fernandes, 2004]. Como o SW-CMM foi evoluído e substituído pelo CMMI (*Capability Maturity Model Integration)* o mercado que exigia o SW-CMM passou a exigir o CMMI.

A seguir será apresentada uma breve descrição sobre o Modelo de Referência da Melhoria de Processo do Software Brasileiro (MR-MPS.BR), modelo de qualidade que o Instituto Centro-Oeste de Desenvolvimento de Software está almejando alcançar com a ajuda do *Personal Software Process*.

4. MELHORIA DE PROCESSO DO SOFTWARE BRASILEIRO – MPS.BR

O MPS.BR – Melhoria de Processo do Software Brasileiro é uma iniciativa da Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), iniciado em dezembro de 2003, e realizado em parceria com diversas instituições de ensino, pesquisa, centros tecnológicos e sociedades de economia mista, visando a melhoria de processo do software Brasileiro, através do desenvolvimento e aprimoramento de um Modelo de Referência para a melhoria do Processo de Software Brasileiro (MR-MPS.BR), compatível com o CMMI (*Capability Maturity Model Integration*) e em conformidade com as normas ISO/IEC 12207 e ISO/IEC 15504; e da implementação e avaliação do Modelo MR-MPS.BR, a um custo acessível em todas as regiões do país, com foco em grupos de pequenas e médias empresas [Weber, 2005].

O MPS.BR baseia-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos. Dentro desse contexto, o MPS.BR possui três componentes [SOFTEX, 2005]:

1) um guia geral com a descrição geral do MPS.BR e um detalhamento do Modelo de Referência (MR-MPS), seus componentes e as definições comuns necessárias para seu entendimento e aplicação;
2) um guia de aquisição, contendo recomendações para a condução de compras de software e serviços correlatos, descrito como uma forma de apoiar as instituições que queiram adquirir produtos de software e serviços correlatos apoiando-se no MR-MPS;
3) um guia de avaliação com a descrição do processo de avaliação, os requisitos para o avaliador, os requisitos para a avaliação, o método e os formulários para apoiar a avaliação.

O Modelo de Referência MPS.BR define sete níveis de maturidade, que são uma combinação entre processos e capacidade de processos, estabelecendo patamares de evolução de processos, caracterizando estágios de melhoria de implementação de processos na organização. Os níveis de maturidade são: 1) em otimização (A); 2) gerenciado quantitativamente (B); 3) definido (C); 4) largamente definido (D); 5) parcialmente definido (E); 6) gerenciado (F); e 7) parcialmente gerenciado (G). A escala de maturidade se inicia no nível G e progride até o nível A [SOFTEX, 2006].

Para cada um destes sete níveis de maturidade é atribuido um perfil de processos que indicam onde a organização deve colocar o esforço de melhoria no processo de desenvolvimento de software. O progresso e o alcance de um determinado nível de maturidade MPS se obtém quando a organização atende os propósitos e todos os resultados esperados dos respectivos processos e dos atributos de processo estabelecidos para aquele nível [SOFTEX, 2006]. A figura 02 apresenta os níveis de maturidade e os respectivos processos do Modelo de Referência MPS.BR.

A seguir será apresentada uma breve descrição sobre o *Personal Software Process*, modelo de qualidade que o Instituto Centro-Oeste de Desenvolvimento de Software está empregando para auxiliá-lo na obtenção do nível G (Parcialmente

Gerenciado) do Modelo de Referência da Melhoria de Processo do Software Brasileiro.

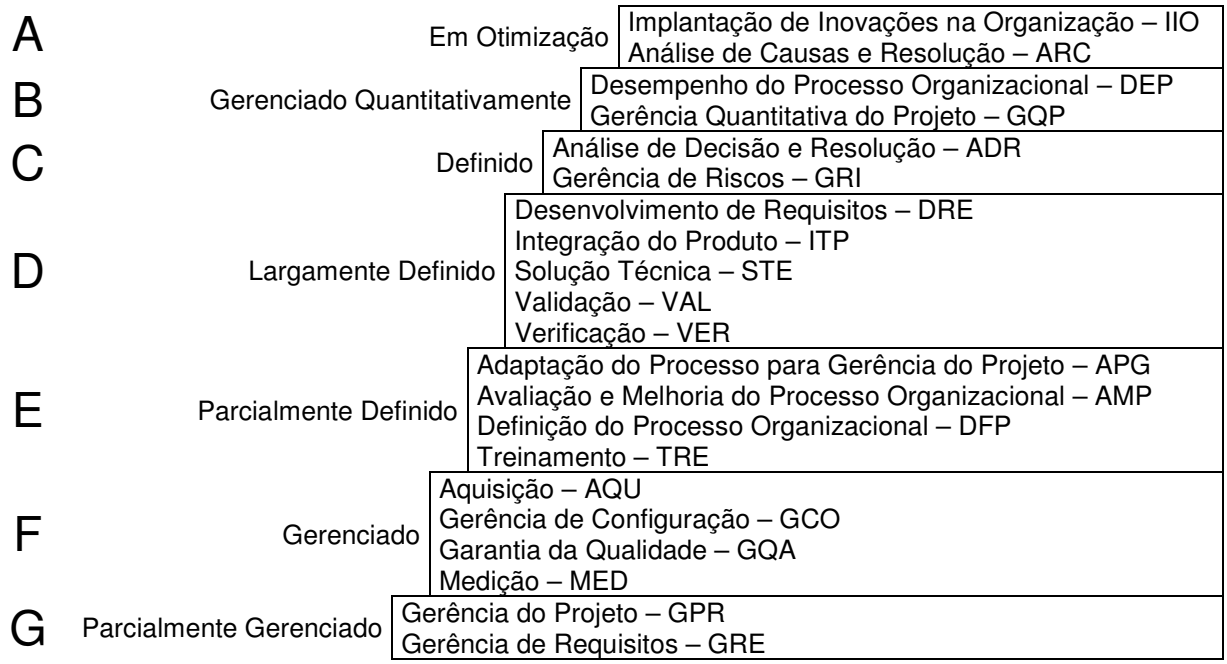

Figura 02 – Níveis de maturidade e processos do Modelo de Referência MPS.BR

5. PERSONAL SOFTWARE PROCESS

O *Personal Software Process* (PSP), traduzido como Processo Pessoal de Software, foi desenvolvido por Watts Humphrey em 1989, apoiado pelo *Software Engineering Institute* (SEI). Humphrey é mundialmente conhecido por desenvolver o processo do *Capability Maturity Model* (CMM) e o *Team Software Process* (TSP) [Koscianski, 2006].

O objetivo do *Personal Software Process* é auxiliar desenvolvedores e pequenas equipes de desenvolvimento de software em suas necessidades de melhoria, tornando o trabalho mais produtivo, adequado e satisfatório ao desenvolvimento de sistemas em escala individual, fazendo com que o Engenheiro de Software encontre seus limites [Humphrey, 2000].

O *Personal Software Process* é uma estrutura de formulários, diretrizes e procedimentos para o desenvolvimento de software, que corretamente usado, provê os dados históricos que o engenheiro de software precisa para fazer e conhecer melhor seus compromissos, e tornar os elementos rotineiros de seu trabalho mais previsíveis e mais eficientes [Silva Júnior, 2000].

O modelo proposto por Humphrey concentra-se no projeto, codificação e fases de testes do desenvolvimento de software. Porém, o *Personal Software Process* não se aplica somente a eles, mas as outras fases do processo de desenvolvimento de software, inclusive na especificação de requisitos, manutenção de produto, planejamento de testes e desenvolvimento de documentação [Hayes, 1997].

Baseado nas mesmas práticas industriais encontradas no *Capability Maturity Model for Software*, a implantação do *Personal Software Process* consiste de quatro etapas, sendo que as três primeiras etapas são subdivididas em dois níveis, totalizando sete níveis de maturidade, de maneira que a implantação do modelo seja

feita de forma incremental, minimizando o impacto na mudança no processo do engenheiro. A figura 03 apresenta os sete níveis de maturidade do *Personal Software Process* [Albuquerque, 2004].

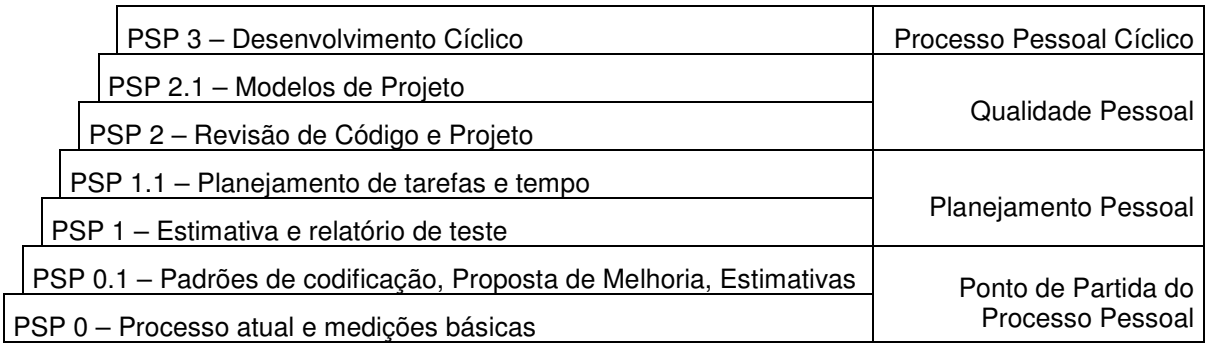

Figura 03 – Níveis de maturidade do *Personal Software Process*

No nível de maturidade PSP 0, os engenheiros introduzem formulários e procedimentos propostos pelo *Personal Software Process* ao seu trabalho pessoal, com o intuito de medir o tempo gasto em cada fase do desenvolvimento, bem como os defeitos inseridos e os encontrados em cada fase, de forma a construir uma base de dados, padronizada, para ser utilizada nos demais níveis de maturidade [Koscianski, 2006].

Dentro do nível de maturidade PSP 0 existe o nível de maturidade PSP 0.1, cujo objetivo é formar um consenso entre os desenvolvedores na adoção de um padrão de codificação para os programas, de modo a tornar uniforme os programas produzidos [Hayes, 1997].

Com a base de dados obtida com o nível de maturidade PSP 0, o nível de maturidade PSP 1 se preocupa em fornecer ao desenvolvedor relatórios de teste, tamanho de código e estimativa para os recursos necessários, ajudando o Engenheiro de Software a assumir compromissos que possa cumprir, fornecendo um planejamento ordenado das tarefas a serem cumpridas e dados para avaliação do trabalho realizado [Humphrey, 2000].

Da mesma forma que no nível de maturidade PSP 0, o nível de maturidade PSP 1 possui o nível de maturidade PSP 1.1, no qual são utilizados alguns métodos para se estimar o tamanho do código, como o *Proxy-Based Estimating*, e também para estimar e escalonar os recursos, como o método de Regressão Linear [Albuquerque, 2004].

O nível de maturidade PSP 2 trata do processo de administração da qualidade pessoal, onde são introduzidas as revisões de código e de projeto ao processo do desenvolvedor, com o estabelecimento de objetivos e guiado por um processo de revisão definido, onde métricas são coletadas de modo a poder avaliar o processo utilizado [Koscianski, 2006].

Já o nível de maturidade PSP 2.1, pertencente ao nível de maturidade PSP 2, estabelece critérios de perfeição de projeto e examina várias técnicas de verificação e consistência de projeto [Humphrey, 2000].

O objetivo do nível de maturidade PSP 3, último nível de maturidade do *Personal Software Process*, é auxiliar o desenvolvedor a lidar com projetos maiores, uma vez que os níveis de maturidade anteriores são baseados em um processo linear para o desenvolvimento de pequenos projetos. A estratégia utilizada é dividir o

projeto em partes menores e aplicar o processo do nível de maturidade PSP 2.1 em cada uma das partes [Hayes, 1997].

6. SOFTWARE LIVRE/CÓDIGO ABERTO (SL/CA)

O movimento de publicação de Software Livre, também conhecido como de Código Aberto (*Open Source*), ganhou notoridade nos últimos anos. Este modo de produção de software tem resultado em produtos de excelente qualidade e grande penetração em alguns setores do mercado mundial de software. A característica mais importante do software livre/código aberto é a liberdade de uso, cópia, modificações e redistribuição, o que lhe confere uma série de vantagens sobre o software proprietário, dentre as quais a sua transformação em bem público, disponível para utilização por toda a comunidade e da maneira que seja mais conveniente a cada indivíduo [Hexsel, 2002].

A liberdade para utilizar e, principalmente, evoluir idéias alheias é um tópico discutido já a um longo tempo pela humanidade, conforme podemos observar na carta de Thomas Jefferson endereçada a Isaac McPherson: "*If nature has made any one thing less susceptible than all others of exclusive property, it is the action of the thinking power called an idea, which an individual may exclusively possess as long as he keeps it to himself; but the moment it is divulged, it forces itself into the possession of every one, and the receiver cannot dispossess himself of it*." [Salus, 2005].

O impulso inicial para a história do software livre/código aberto foi dado em 1969, quando Ken Thompson desenvolveu a primeira versão do sistema operacional *Unix*, o qual era distribuído gratuitamente para as universidades e centros de pesquisa. Mas em 1975, o sistema operacional *Unix* começou a ser controlado pelo *Unix System Laboratories* (USL), uma subsidiária dos laboratórios Bell com objetivos comerciais, de forma que as pessoas e organizações deveriam agora pagar uma taxa a USL pela utilização do sistema operacional *Unix* [Alves, 2005].

Em 1983 Richard Stallman, que até então trabalhava no *Massachusets Institute of Technology* (MIT), fundou a *Free Software Fundation* (FSF) com o objetivo de promover na comunidade de informática o espírito cooperativo que prevalecia em seus primórdios, onde as informações, códigos e métodos de trabalho eram livremente compartilhados [Basic, 2003].

A FSF define como software livre/código aberto aquele que qualquer um pode executar, copiar, distribuir, estudar, modificar e aperfeiçoar. Mas precisamente, o conceito de software livre requer que os termos de distribuição do programa ofereçam estas quatro liberdades aos seus usuários [Alves, 2005]:

- A liberdade de executar o programa, para qualquer propósito;
- A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades. O acesso ao código fonte é um pré-requisito para esta liberdade;
- A liberdade de redistribuir cópias de modo a ampliar as possibilidades de acesso de pessoas e instituições a tais programas;
- A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie, sem gastos adicionais.

7. IMPLANTAÇÃO DO PERSONAL SOFTWARE PROCESS

O projeto de implatanção do *Personal Software Process* nas dependências do Instituto Centro-Oeste de Desenvolvimento de Software foi iniciado em maio de

2006, com a criação de um grupo de trabalho, composto por três colaboradores em tempo parcial, responsáveis pelo estudo e análise da metodologia selecionada.

Entre as atribuições do grupo de trabalho estavam a definição do escopo a ser utilizado na organização, a elaboração das planilhas de coleta de dados, realização dos treinamentos e a implementação de uma ferramenta *Web open source* para auxiliar e automatizar o processo de coleta e análise dos dados provenientes do *Personal Software Process*.

O grupo de trabalho decidiu começar a implantação do *Personal Software Process* de forma gradual, de modo que num primeiro momento, seriam implantadas os níveis de maturidade PSP 0 e PSP 0.1, deixando os demais níveis para serem implementados posteriormente, ao fim da implantação dos dois primeiros níveis de maturidade.

Após a definição do escopo de implantação do *Personal Software Process*, o grupo de trabalho se concentrou na elaboração das planilhas a serem utilizadas pelos colaboradores para o registro das suas atividades, e principalmente, no treinamento e conscientização dos colaboradores para a utilização da nova metodologia a ser empregada.

As informações pertinentes ao *Personal Software Process* foram coletadas pelos colaboradores através de planilhas eletrônicas, conforme exibida na figura 04, onde cada colaborador deve registrar as tarefas realizadas no decorrer do dia, bem como o esforço (em horas) despendido para a realização de cada tarefa.

Data	***Início***	***Atividade***	***Projeto***	***Descrição***	***Concluído***	***Interrupção***	***Fim***
02	08:00	1.1.1	OBID06	Apresentação do serviço	Sim	00:00	09:30
02	09:30	1.1.2	OBID06	Análise do serviço	Sim	00:05	10:00
02	10:00	1.1.3	OBID06	Elaboração do Termo de Abertura	Sim	00:15	12:00
02	12:00	1.1.4	OBID06	Elaboração da Proposta Técnica	Não	00:20	18:00
02	08:00	1.1.4	OBID06	Elaboração da Proposta Técnica	Sim	00:15	12:00

Figura 04 – Planilha de coleta de informações para o *Personal Software Process*

Conforme observado na figura 04, a planilha de coleta de dados é composta por oito campos, descritos da seguinte forma:

- data: dia da realização da atividade pelo colaborador;
- início: horário de início da execução da atividade pelo colaborador, no formato hora:minuto (hh:mm);
- atividade: código da atividade realizada pelo colaborador;
- projeto: código do projeto no qual o colaborador está executando a atividade;
- descrição: detalhamento da atividade executada pelo colaborador;
- concluído: indicar se a atividade executada foi completamente concluída;
- interrupção: tempo despendido com interrupções na realização da atividade, no forma hora:minuto (hh:mm), como por exemplo, atendimento de telefonemas, pausa para descanso, entre outras interrupções;
- fim: horário de finalização da execução da atividade pelo colaborador, no formato hora:minuto (hh:mm).

Conforme mencionado anteriormente, o Instituto Centro-Oeste de Desenvolvimento de Software está adequando os seus processos ao modelo de referência da Melhoria do Processo de Software Brasileiro, de forma que a maioria das atividades realizadas pelos colaboradores durante a execução de um projeto são mapeadas e descritas no seu processo de desenvolvimento.

Cada atividade do processo de desenvolvimento possui um código, de forma que os colaboradores não precisam exemplificar a atividade realizada no momento, mas apenas indicar o seu código na planilha de coleta de dados do *Personal Software Process*.

As informações provenientes das planilhas eletrônicas dos colaboradores são consolidadas semanalmente, visando serem utilizadas pela organização na elaboração de estimativas confiáveis e reais para o planejamento do esforço e do custo dos produtos de trabalho e tarefas a serem executados nos novos projetos. A figura 05 apresenta o esboço da planilha consolidada.

Atividade	***OBID06***	***GPR06***	***GRE06***	***MÉDIA***
1. Fase de Concepção				
1.1. Solicitação de Serviço	5,0	7,9	3,6	5,5
1.1.1. Reunião com o cliente	1,5	2,5	1,1	1,7
1.1.2. Avaliação da solicitação	0,5	0,9	0,4	0,6

Figura 05 – Consolidação das informações do *Personal Software Process*

As informações puderam ser consolidadas, para servirem de base para as estimativas dos projetos futuros devido a construção de um processo de desenvolvimento padrão, de forma que os projetos realizados pela organização seguem uma instância do processo de desenvolvimento padrão.

O padrão de codificação empregado pelo Instituto Centro-Oeste de Desenvolvimento de Software para a padronização do código produzido pelos seus colaboradores, segue as diretrizes do padrão de codificação da linguagem Java da *HotWork Solution* [HotWork, 2004].

Uma vez estabelecido o *Personal Software Process* entre os colaboradores, o grupo de trabalho começou a implementação de um sistema *Web open source*, desenvolvido sobre a plataforma J2EE (*Java 2 Enterprise Edition*), para automatizar o registro e a consolidação das informações provenientes dos colaboradores.

8. FERRAMENTA WEB OPEN SOURCE PARA A IMPLANTAÇÃO DO PSP

O objetivo da ferramenta *Web open source* para a implantação do *Personal Software Process* é auxiliar os colaboradores do Instituto Centro-Oeste de Desenvolvimento de Software no registro das atividades executadas pelos colaboradores, mas principalmente, automatizar o processo de consolidação das informações provenientes dos diversos colaboradores, de forma que os registros possam ser utilizados como dados históricos.

Para que os registros fossem utilizados como base para as estimativas dos futuros projetos, o processo padrão do Instituto Centro-Oeste de Desenvolvimento de Software deve estar de alguma forma registrado na ferramenta, de modo que as atividades executadas pelos colaboradores estejam atreladas ao processo padrão. O processo padrão do Instituto Centro-Oeste de Desenvolvimento de Software pode ser decomposto em quatro níveis: fases, processos, atividades e tarefas.

Atualmente, o processo de desenvolvimento de software padrão do Instituto Centro-Oeste de Desenvolvimento de Software é composto por cinco fases, sendo elas: iniciação, concepção, elaboração, construção e transição.

Cada fase pode conter um número ilimitado de processos, como por exemplo, a fase de iniciação pode conter os processos de Solicitação de Serviço, Elaboração do Termo de Abertura do Projeto, Realização da Prospecção Inicial, Elaboração da Proposta Técnica, Elaboração da Proposta Comercial, entre outros processos.

E cada processo, pode conter um número ilimitado de atividades, com por exemplo, o processo de Solicitação de Serviço pode conter as atividades de Reunião Inicial com o Cliente, Avaliação da Solicitação, entre outras atividades.

Por fim, cada atividade pode ser composta por um número ilimitado de tarefas, onde o processo de desenvolvimento de software padrão é instanciado para cada projeto. Cada tarefa a ser executada no projeto deve estar alocada a um colaborador, que é um usuário do sistema e que, no referido projeto, desempenha um determinado papel, como por exemplo, analista de sistemas, programador, testador, entre outros papéis.

Com a decomposição do processo padrão de desenvolvimento de software em vários níveis, o sistema consegue apresentar aos coordenadores de projeto e aos diretores da organização, o esforço despendido pela organização na execução de um determinado projeto, em vários níveis de refinamento (fases, processos e atividades).

A figura 06 apresenta o modelo entidade-relacionamento do sistema proposto, onde é possível visualizar mais claramente a decomposição do processo padrão de desenvolvimento de software do Instituto Centro-Oeste de Desenvolvimento de Software, bem como a forma como as atividades são relacionadas com os colaboradores e projetos.

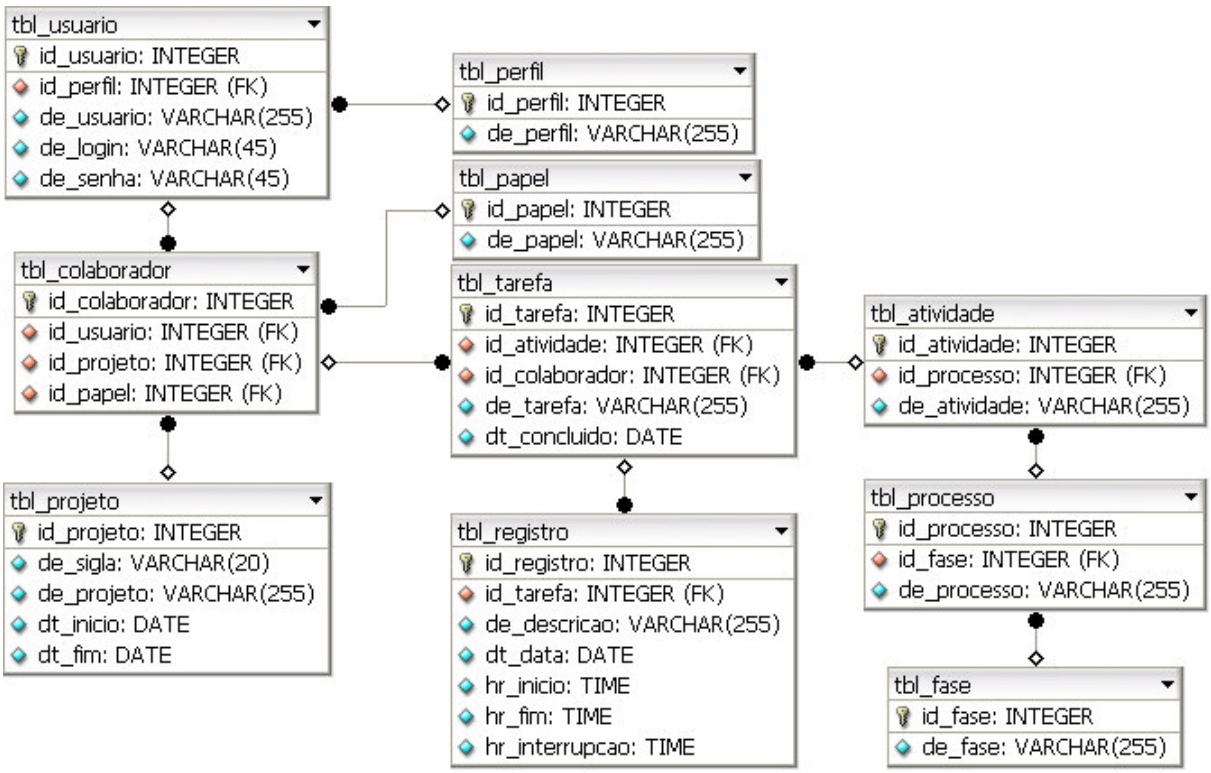

Figura 06 – Modelo Entidade-Relacionamento

É de responsabilidade do grupo de processos a definição e a administração do processo padrão de desenvolvimento de software, sendo os únicos autorizados pelo sistema a editarem as informações de fases, processos e atividades.

Ao gerente do projeto é atribuída a responsabilidade de instanciar o processo padrão ao projeto, definindo quais atividades serão executadas no projeto, e alocando as mesmas aos colaboradores que participarão da execução do projeto.

Ao colaborador, é atribuída a responsabilidade de registrar seus esforços para a conclusão de cada uma das atividades sob sua competência, fornecendo as informações que serão utilizadas posteriormente para formar a base de dados histórica.

A figura 07 apresenta a interface onde o colaborador inicia a execução de uma nova tarefa. Nesse caso, o colaborador deve informar ao sistema o projeto, seu papel no projeto, fase, processo, atividade e tarefa que irá executar a partir daquele momento, bem como uma breve descrição do que irá fazer, caso necessário.

Figura 07 – Interface para o registro do início da tarefa

A necessidade do usuário selecionar seu papel dentro do projeto se deve ao fato de que um determinado usuário pode exercer vários papéis durante a execução de um determinado projeto.

Quando o colaborador encerrar a execução da presente tarefa, o mesmo deverá confirmar se a descrição da tarefa informada anteriormente está correta, o período gasto com interrupções e se a tarefa foi concluída ou não, conforme apresentada na figura 08.

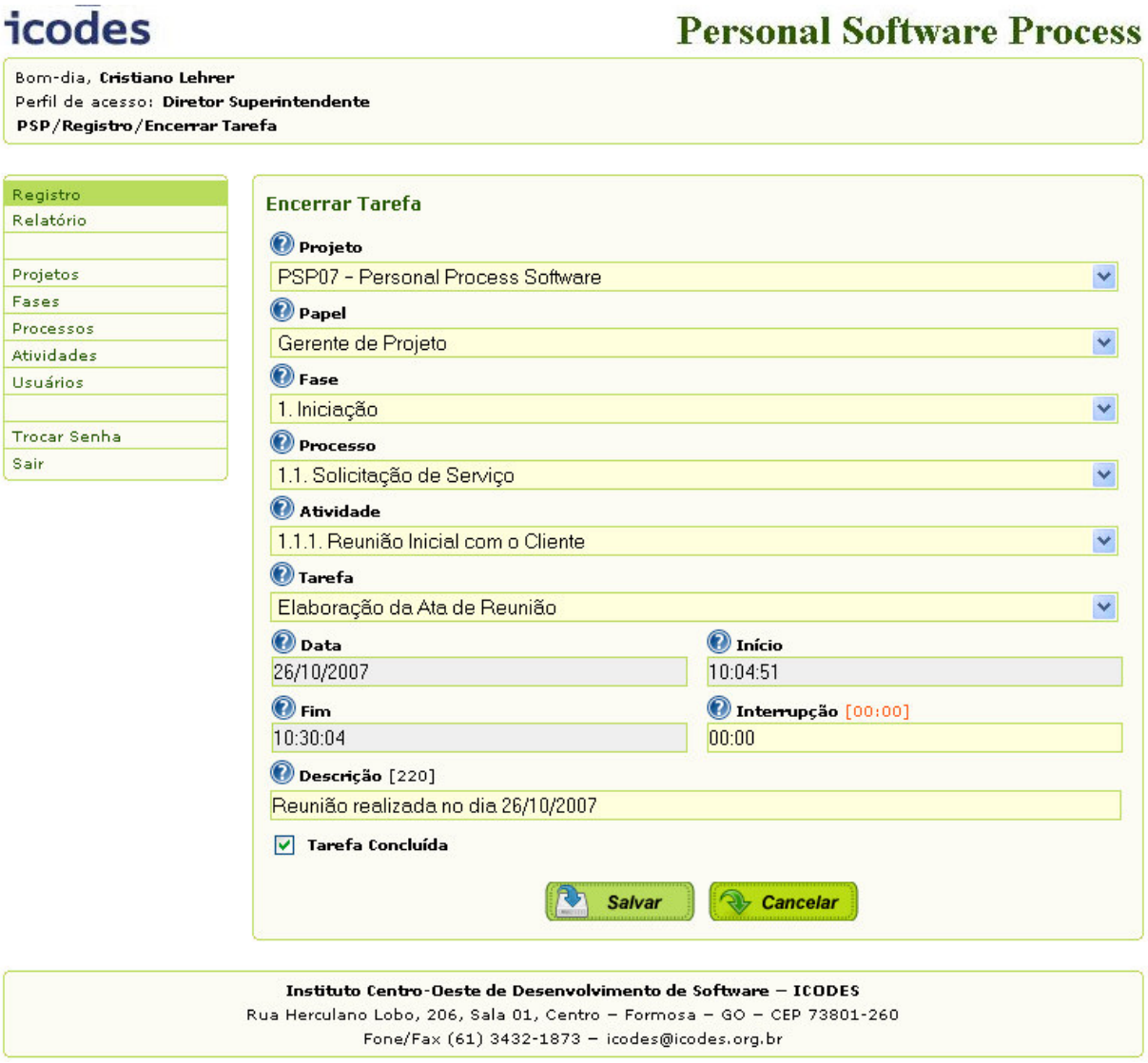

Figura 08 – Interface para o registro do encerramento da tarefa

A tecnologia selecionada para o desenvolvimento da ferramenta *Web open source* para auxiliar na implantação do *Personal Software Process* nas dependências do Instituto Centro-Oeste de Desenvolvimento de Software foi a tecnologia *Java 2 Enterprise Edition* (J2EE), da Sun Microsystems, que é um padrão dinâmico para a produção de sistemas de informação corporativos seguros, escaláveis e independentes de plataforma [Bond, 2003].

A escolha da plataforma J2EE também determinou a seleção das demais ferramentas e tecnologias a serem adotadas na implementação da ferramenta. Dessa forma, o paradigma utilizado para o desenvolvimento é o orientado a objetos [Barnes, 2004], tendo como linguagem de modelagem a *Unified Modelling Language* (UML) [Booch, 2000] e como linguagem de codificação o Java 2 [Deitel, 2003]. A tabela 01 apresenta o ambiente computacional utilizado para o desenvolvimento da referida ferramenta.

Tabela 1 – Ambiente computacional

Categoria	Software
Ambiente Integrado de Desenvolvimento	Eclipse SDK 3.2.2
Banco de Dados	MySQL 4.0.20
Biblioteca	API Servlet 2.3
Biblioteca	Connector/J 5.0.6
Biblioteca	Log4J 1.2.14
Biblioteca	Java Server Pages 1.2
Comunicador	Skype 3.2.0.158
Ferramenta de Banco de Dados	MySQL Administrator 1.2.8
Ferramenta de Banco de Dados	MySQL Query Browser 1.2.8
Ferramenta de Escritório	BrOffice.org 2.2.0
Linguagem de Programação	Java 2 Plataform, Standard Edition 1.4.2
Modelagem dos Dados	DBDesigner 4.0.5.6
Modelagem UML	ArgoUML 0.24
Navegador de Internet	Mozilla Firefox 2.0.0.8
Servidor de Aplicação Web	Apache Tomcat 4.1.36
Sistema Operacional	Fedora Core 5

Para auxiliar e agiliar o processo de desenvolvimento da ferramenta, foi desenvolvido e implementado uma arquitetura de desenvolvimento padrão para os componentes de software, de modo que cada componente de software possuísse um conjunto básico de classes e interfaces a serem implementadas. A figura 09 apresenta a arquitetura de desenvolvimento utilizada no desenvolvimento da ferramenta.

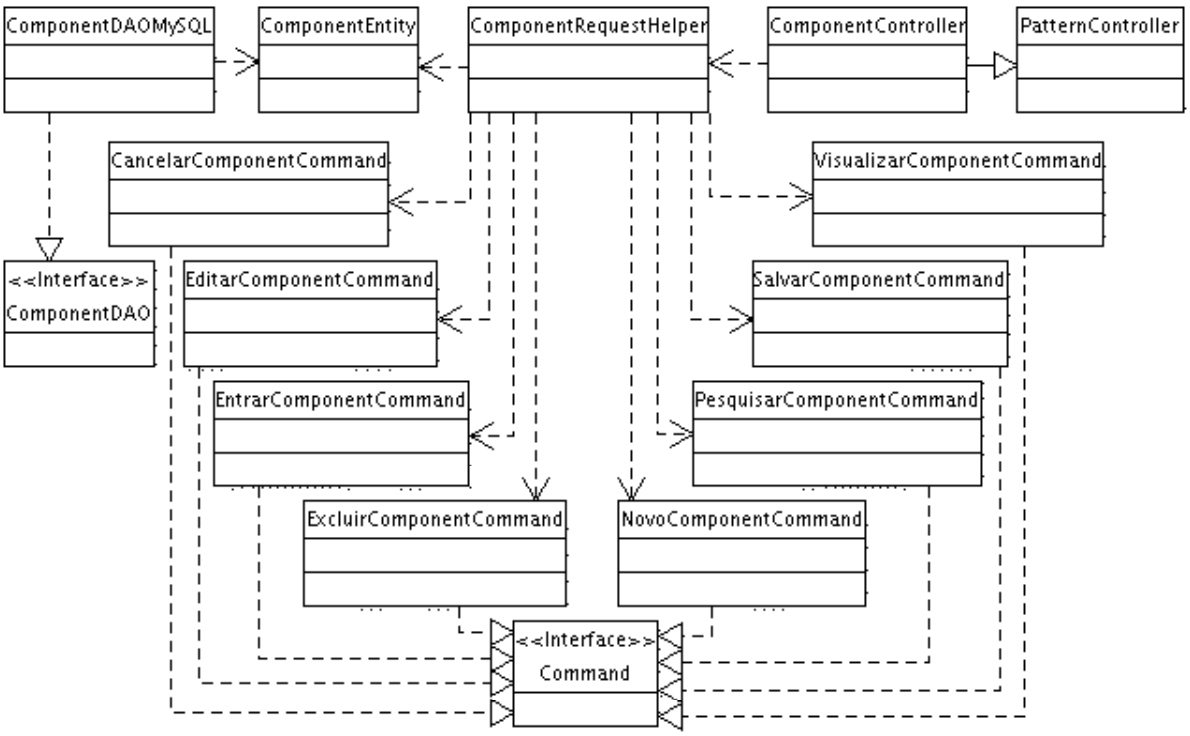

Figura 09 – Arquitetura de Desenvolvimento

A arquitetura de desenvolvimento exibida na figura 09 apresenta apenas as principais classes dos padrões de projetos utilizados no desenvolvimento do projeto. Nessa arquitetura em específica, estavam sendo utilizados os padrões de projeto *Intercepting Filter*, *Front Controller*, *View Helper*, *Composite View*, *Value Object* e *Data Access Object* de Alur [2002], e os padrões de projeto *Abstract Factory*, *Singleton* e *Command* de Gamma [2000].

O processo de desenvolvimento de software adotado na realização do presente projeto é baseado na metodologia de processo *lightweight* chamada *Extreme Programming* (XP), visando alcançar um desenvolvimento rápido e consistente com as reais necessidades do Instituto Centro-Oeste de Desenvolvimento de Software, permitindo que o software seja modificado à medida que as necessidades do negócio se alteram ou se ampliam.

O processo de desenvolvimento utilizado é composto por seis etapas, repetidas para cada componente desenvolvido. As etapas do processo de desenvolvimento de software são:

- Especificação dos Requisitos: levantamento das funcionalidades a serem implementadas no componente;
- Especificação dos Testes Funcionais: elaboração dos casos de teste do componente;
- Especificação do Projeto: elaboração do projeto do componente, englobando a especificação do diagrama de classes e do modelo entidade-relacionamento (MER);
- Codificação do Componente: implementação do componente;
- Realização dos Testes Funcionais: execução dos casos de teste sobre o componente implementado;
- Integração do Componente: inclusão do componente desenvolvido ao restante da aplicação.

A figura 10 apresenta o fluxo do processo de desenvolvimento de software adotado. As linhas contínuas representam o fluxo normal de trabalho do processo de desenvolvimento, enquanto que as linhas pontilhadas representam as exeções, no qual o processo deve retornar para uma atividade anterior para sua correta finalização.

9. CONCLUSÃO E PERSPECTIVAS FUTURAS

A indústria de software vem experimentando um grande crescimento nas últimas décadas. Atualmente, o software está presente na vida de praticamente todas as pessoas, seja através do uso de computadores, sistemas de automação comercial e industrial, ou de software embutido em eletrodomésticos e telefones celulares, fazendo com que o mercado exigisse produtos de qualidade a um baixo custo [Vasconcelos, 2005].

Segundo pesquisas realizadas, para cada real investido em aprimoramentos de processos de software pode trazer um retorno de até sete reais, ou seja, qualidade em software sai barato [Couto, 2007].

Comparado com outros processos de melhoria da qualidade, como a ISO 9000, CMMI ou o próprio MPS.BR, a implantação do *Personal Software Process* requer um investimento significativamente menor, proporcionalmente aos benefícios e retornos esperados, permitindo que seja utilizado por micro e pequenas empresas

produtoras de software, que não dispõem de muitos recursos para serem aplicados internamente.

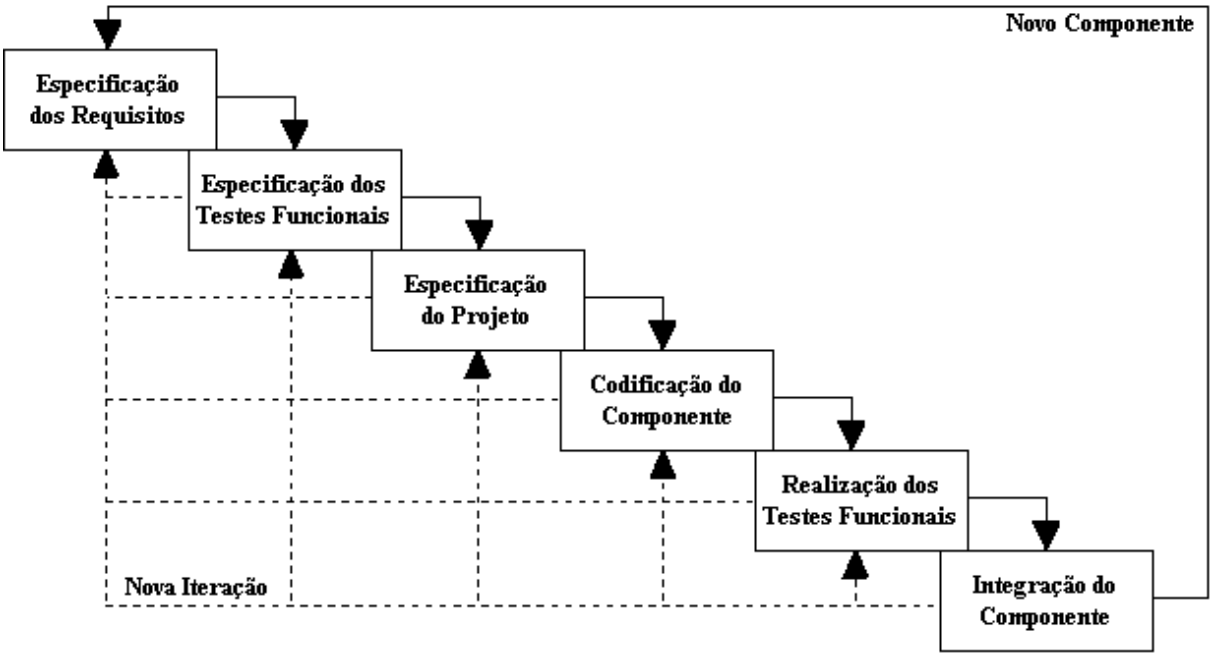

Figura 10 – Processo de Desenvolvimento de Software

A implantação do *Personal Software Process* exigiu um esforço considerável da organização, e principalmente, dos seus colaboradores, não apenas ao executar o trabalho e ao estudar os próprios métodos, mas principalmente na interferência com as práticas atuais.

A utilização do *Personal Software Process* pelo Instituto Centro-Oeste de Desenvolvimento de Software contribuiu para a melhoria das estimativas utilizadas para a elaboração do Plano de Projeto, melhorando significativamente o planejamento e o acompanhamento dos cronogramas dos projetos. Por exemplo, no início do referido projeto, a previsão para o desenvolvimento de cada módulo do sistema era de três semanas (quinze dias), mas pelas informações coletadas até o momento, descobriu-se que os desenvolvedores levam na verdade dez dias, em média, para o desenvolvimento dos módulos.

Dessa forma, os novos projetos em andamento e/ou negociação no Instituto Centro-Oeste de Desenvolvimento de Software já consideram que o tempo médio para o desenvolvimento (planejamento, implementação e teste) de um módulo de sistema é de dez dias em média, facilitando a negociação dos prazos finais de entrega perante os clientes.

Atualmente, o Instituto Centro-Oeste de Desenvolvimento de Software está implantando seu sistema computadorizado para facilitar e agilizar o preenchimento dos formulários requeridos pelo *Personal Software Process*, de forma que o desenvolvedor não desperdice tempo preenchendo os formulários e gerando as estatísticas, além de automatizar as tarefas de consolidação dos registros para a organização, fornecendo um retorno mais rápido e preciso sobre o desempenho dos desenvolvedores.

A conclusão definitiva do projeto está prevista para ocorrer no primeiro trimestre de 2008, uma vez que os colaboradores se encontram elaborando os diversos relatórios de consolidação das informações.

Assim que a ferramenta estiver totalmente implantada e em operação, o Instituto Centro-Oeste de Desenvolvimento de Software pretende começar a adequar a ferramenta desenvolvida para suportar os outros níveis de maturidade do *Personal Software Process*, buscando sempre alcançar níveis mais elevados de qualidade e produtividade.

A partir do primeiro trimestre de 2008, a ferramenta estará disponível para toda a comunidade, podendo ser adquirida livremente através da *home page* do Instituto Centro-Oeste de Desenvolvimento de Software.

10. AGRADECIMENTOS

Os autores agradecem ao Instituto Centro-Oeste de Desenvolvimento de Software que viabilizou a execução desse trabalho.

11. REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, Jones de Oliveira. (2004). *Gerencia de Processo de Software em Pequena Escala*: ênfase em PSP, TSP e P-CMM. Lavras: UFLA/FAEPE. 113p.

ALUR, Deepak. (2002). *Core J2EETM patterns*: as melhores práticas e estratégias de design. Rio de Janeiro: Campus.

ALVES, Ângela Maria. (2005). *Introdução à Produção de Software*: Software Livre / Código Aberto (SL/CA). Lavras: UFLA/FAEPE.

BARNES, David J.; KÖLLING, Michael. (2004) *Programação Orientada a Objetos com Java*. São Paulo: Pearson Prentice Hall.

BASIC, Nicolas Michael. (2003). *O software livre como alternativa ao aprisionamento tecnológico imposto pelo software proprietário*. Disponível em: <http://www.rau-tu.unicamp.br/nou-rau/softwarelivre/document/?down=107>. Acesso em: out. 2007.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. (2000). *UML*: guia do usuário. Rio de Janeiro: Campus.

BOND, Martin; [et al.] (2003) *Aprenda J2EE com EJB, JSP, Servlets, JNDI, JDBC e XML*. São Paulo: Pearson Education do Brasil.

CÉSAR, Ricardo. (2003). *Fábrica de Software*: uma vocação nacional? Disponível em: <http://www.siscorp.com.br/imprensa/computerworld02.htm?documento=24655&Area=51>. Acesso em: mar. 2007.

COUTO, Ana Brasil. (2007). *CMMI*: integração dos modelos de capacitação e maturidade de sistemas. Rio de Janeiro: Editora Ciência Moderna.

DEITEL, H. M.; DEITEL, P. J. (2003). *Java, como programar*. 4ª Edição. Porto Alegre: Bookman.

FERNANDES, Aguinaldo Aragon; TEIXEIRA, Descartes de Souza. (2004). *Fábrica de Software*: implantação e gestão de operações. São Paulo: Atlas. 304p.

GAMMA, Erich. (2000). *Padrões de projeto*: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman.

HAYES, Will; OVER, James W. (1997). *The Personal Software Process^{SM} (PSP^{SM})*: an empirical study of the impact of PSP on individual engineers. Pittsburgh: SEI. 76p. Disponível em: <http://www.sei.cmu.edu/pub/documents/97.reports/pdf/97tr001.pdf>. Acesso em: mar. 2007.

HERBSLEB, J. D.; GRINTER, R. E. (1999). Splitting the Organization and Integrating the Code: Conway's Law Revisited. In: *Proceedings of ICSE*. Los Angeles: IEEECSP. Pág. 85-95.

HEXSEL, Roberto A. (2002). Propostas de ações de governo para incentivar o uso de software livre. *Relatório técnico do departamento de informática da UFPR*, 004/2002, out02.

HOTWORK Solution. (2004). *Java Programming Guidelines*. Disponível em: <http://hotwork.sourceforge.net/hotwork/manual/guidelines/code-guidelines-user-guide.html>. Acesso em: mar. 2007.

HUMPHREY, Watts S. (2000). *The Personal Software Process^{SM} (PSP^{SM})*. Pittsburgh: SEI. 55p. Disponível em: <http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr022.pdf>. Acesso em: mar. 2007.

ICODES – Instituto Centro-Oeste de Desenvolvimento de Software. (2006). *Estatuto Social da Sociedade para o Desenvolvimento da Tecnologia da Informação e Comunicação do Centro-Oeste*. Disponível em: <http://www.icodes.org.br>. Acesso em: mar. 2007.

ICODES – Instituto Centro-Oeste de Desenvolvimento de Software. (2007). *Perfil*. Disponível em: <http://www.icodes.org.br>. Acesso em: mar. 2007.

KOSCIANSKI, André; SOARES, Michel dos Santos. (2006). *Qualidade de Software*: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. São Paulo: Novatec Editora. 395p.

MARQUES, H. M. et al. (2003). *Fábricas de Software e o Processo de Desenvolvimento Segundo a Experiência da FábricaUm*. Universidade Federal de Pernambuco. Disponível em: <http://www.cin.ufpe.br/~in953/olds/relatorios/fabrica1.pdf>. Acesso em: mar. 2007.

PRESSMAN, Roger S. (1995). *Engenharia de Software*. 3ª Edição. São Paulo: Makron Books. 1.056p.

ROCHA, Ana Regina Cavalcanti da; MALDONADO, José Carlos; WEBER, Kival Chaves. (2001). *Qualidade de Software*: teoria e prática. São Paulo: Prentice Hall. 303p.

SALUS, Peter H. (2005). *A History of Free and Open Source*. Disponível em: <http://www.groklaw.net/article.php?story=20050327184603969>. Acesso em: out. 2007.

SILVA JÚNIOR, José Wilson da. (2000). *Uma Disciplina para a Engenharia de Software*: Estudo do Personal Software Process (PSP), Proposto por Watts Humphrey (A Profissionalização do Desenvolvedor de Software). 120 f. Monografia (Graduação) – Curso de Bacharelado de Informática, Instituto de Física e Matemática, Universidade Federal de Pelotas, Pelotas.

SOFTEX – Associação para Promoção da Excelência do Software Brasileiro. (2005) *Guia Geral*: MPS.BR – Melhoria de Processo do Software Brasileiro. Disponível em: <http://www.softex.br/media/guia.pdf>. Acesso em: dez. 2005.

SOFTEX – Associação para Promoção da Excelência do Software Brasileiro. (2006) *Curso de Introdução ao MPS.BR* (C1-MPS.BR). SOFTEX.

VASCONCELOS, Alexandre Marcos Lins de. (2005). *Processos de Desenvolvimento de Software*. Lavras: UFLA/FAEPE.

VASCONCELOS, Alexandre Marcos Lins de; MACIEL, Teresa Maria de Medeiros. (2003). *Introdução à Engenharia de Software e aos Princípios de Qualidade*. Lavras: UFLA/FAEPE. 104p.

WEBER, Kival Chaves; ARAÚJO, Eratóstenes; MACHADO, Cristina Ângela Filipak; SCALET, Danilo; SALVIANO, Clênio Figueiredo; ROCHA, Ana Regina Cavalcanti da. (2005). Modelo de Referência e Método de Avaliação para Melhoria de Processo de Software – versão 1.0 (MR-MPS e MA-MPS). In: *IV Simpósio Brasileiro de Qualidade de software*. Porto Alegre: SBQS, Junho de 2005.