WELLERSON LOPES DA SILVA

GERÊNCIA DE PROCESSO: UMA PROPOSTA DE ARQUITETURA DE SOFTWARE DE APOIO AO TSP

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Orientador
Prof.:Jones Oliveira de Albuquerque
Co-Orientadora
Profª:Ana Cristina Rouiller

LAVRAS
MINAS GERAIS – BRASIL
2001

WELLERSON LOPES DA SILVA

GERÊNCIA DE PROCESSO: UMA PROPOSTA DE ARQUITETURA DE SOFTWARE DE APOIO AO TSP

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 26 de junho de 2001.

_____
Prof. Antônio Maria Pereira de Resende

_____
Profª. Ana Cristina Rouiller
(Co-orientadora)

_____
Prof. Jones Oliveira de Albuquerque
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL

DEDICATÓRIA

A Deus, ao meu orientador Jones Oliveira de Albuquerque e co-orientadora Ana Cristina Rouiller.

AGRADECIMENTOS

A empresa Devex Tecnologia e Sistemas por ser o “laboratório” de meus experimentos. A meus pais que não mediram esforços em ajudar-me. A minha noiva, pelos incentivos.

RESUMO

Esta monografia apresenta uma proposta de arquitetura de software de apoio ao modelo de gerência de processo TSP, *Team Software Process*. A arquitetura é uma adaptação do *ProjectSpace*, um projeto de iniciativa do Centro de Informática da Universidade Federal do Pernambuco, do Centro de Estudos Avançados de Recife, da Universidade Federal de Lavras e da empresa Devex Tecnologia e Sistemas Ltda. O *ProjectSpace* objetiva gerenciar os projetos de software de modo a obter métricas para a melhoria do processo de desenvolvimento. Apresenta também conclusões quanto a validação do protótipo do *ProjectSpace*, a saber TimeSheet e PlanEngine, que está em uso na Devex e trabalhos futuros quanto a arquitetura proposta. A validação visa verificar se o protótipo atende aos requisitos do modelo TSP.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

Esta monografia de graduação, apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado, tem o objetivo de atender os requisitos para obtenção do título de Bacharel em Ciência da Computação. Ela aborda a utilização de um modelo de gerência de processo, o TSP, *Team Software Process,* dentro de um projeto entitulado *ProjectSpace*. O *ProjectSpace* é um projeto de iniciativa do Centro de Informática da UFPE em cooperação com o CESAR - Centro de Estudos Avançados de Recife, a Universidade Federal de Lavras e a Devex Tecnologia e Sistemas. Este projeto conta com um grupo de aproximadamente 16 pessoas e tem como principal motivação o fato de que gerenciando os projetos de software, registrando e acompanhando as suas execuções torna-se possível obter métricas para introduzir melhorias no processo de software.

De modo mais específico, este trabalho abordou uma adaptação da arquitetura do *ProjectSpace* de acordo com as necessidades do modelo TSP. Abordou também a validação do protótipo.do *ProjectSpace,* que já vem sendo utilizado pela Devex, frente as reais necessidades do modelo TSP juntamente com conclusões e futuros trabalhos. A proposta desta arquitetura de apoio ao modelo TSP visa atender às necessidades de uma ferramenta que auxilie na implantação deste modelo.

Este trabalho está dividido em 5 capítulos. O Capítulo 2 aborda o papel da engenharia de software frente aos problemas associados à construção e ao desenvolvimento de software. O Capítulo 3 aborda os modelos de gerência de processos, especificadamente o modelo PSP, *Personal Software Process*, e o TSP, *Team Software Proces*. O Capítulo 4 aborda detalhes associados ao uso *Project Space*-TSP. O Capítulo 5 apresenta conclusões e futuros trabalhos.

2 O PAPEL DA ENGENHARIA DE SOFTWARE

Com a evolução do software, os problemas associados à construção e ao desenvolvimento de software se intensificaram e se tornaram mais complexos de serem gerenciados. Alguns fatores contribuíram para este aumento de complexidade: falta de metodologia na construção do software; capacitação inadequada dos desenvolvedores; manutenção dos programas ameaçada por projetos ruins e recursos inadequados; construção de hardware mais poderoso exigindo software que extraísse todo o seu potencial; estimativas de prazos e custos imprecisas; baixa produtividade dos profissionais desenvolvedores de software; qualidade de software inadequada. A engenharia de software fornece métodos, técnicas e ferramentas em resposta a estes problemas.

A engenharia de software é constituída por uma tríplice formada de métodos, ferramentas e procedimentos [PRESSMAN, 1995]. Os métodos são os modelos para a construção de software, as ferramentas fornecem apoio automatizado ou semi-automatizado aos métodos e os procedimentos definem a seqüência de métodos a serem aplicados para a construção do software. Estes três componentes fornecem ao desenvolvedor uma estrutura básica para controlar e gerenciar o processo de desenvolvimento do software e para construir software de qualidade de forma eficiente.

2.1 Métodos

Os métodos utilizados nas fases de desenvolvimento de software são específicos e alguns dependem fortemente da tecnologia utilizada, pois são desenvolvidos para satisfazer um determinado objetivo. Alguns métodos desenvolvidos para um projeto perdem totalmente seu significado se aplicados a outro projeto.

O principal objetivo da engenharia de software é produzir software de alta qualidade a baixo custo [JALOTE, 1992]. O custo pode ser calculado ao final do desenvolvimento ou estimado mediante métodos estatísticos consolidados, por exemplo, COCOMO [BOEHM, 1981] e SLIM [KITCHENHAM et al, 1985], entre outros [MOORE et al, 1976; PILLAI et al,1997; KARLSSON et al, 1997]. Estes métodos são bastante genéricos e fornecem recursos para avaliação quantitativa do processo de desenvolvimento de software.

2.2 Ferramentas

A utilização de ferramentas no processo de software mostra que ou se utilizam planejamento, processos adequados e ferramentas especializadas, ou sem o uso de ferramentas tem-se perda de produtividade, baixa qualidade e custo elevado nas fases de teste e manutenção do software [FLECHER et al, 1993].

As ferramentas possuem a função de automatizar e otimizar os recursos e a realização de tarefas. A automação possui dois componentes básicos:

- mecanização das tarefas existentes; e
- capacidade de realizar tarefas diferenciadamente.

O princípio básico da automação é o rigor e a disciplina de processo, como ilustrado na Figura 2.1.

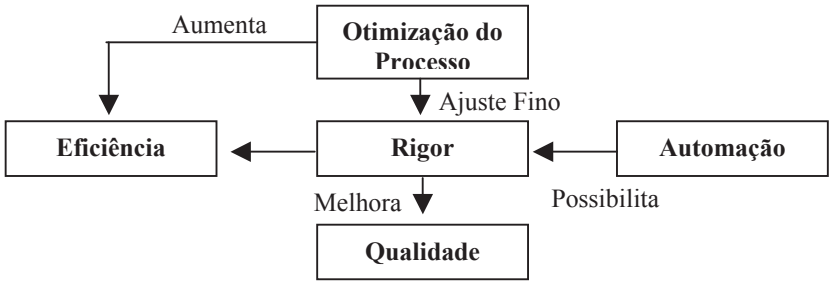

Figura 2.1 - Princípios de Engenharia de Software

Dentre os aspectos de CASE, *Computer Aided Software Engineering*, as ferramentas são os mais discutidos, pois são mais visíveis e de maior custo. São ainda mais visíveis quando não respondem às necessidades do processo de software no qual estão inseridas. A falta de critérios na seleção de uma ferramenta é o principal fator para a utilização inadequada e ineficiente [ZARELLA, 1990].

Ferramentas podem suportar as diversas fases do ciclo de vida do software, e são classificadas de acordo com a sua funcionalidade. Em [SOMMERVILLE, 1992], as ferramentas estão classificadas, ortogonalmente, em dois critérios: orientadas à atividade e orientadas à funcionalidade. Esta classificação pode ser melhor observada na Figura 2.2.

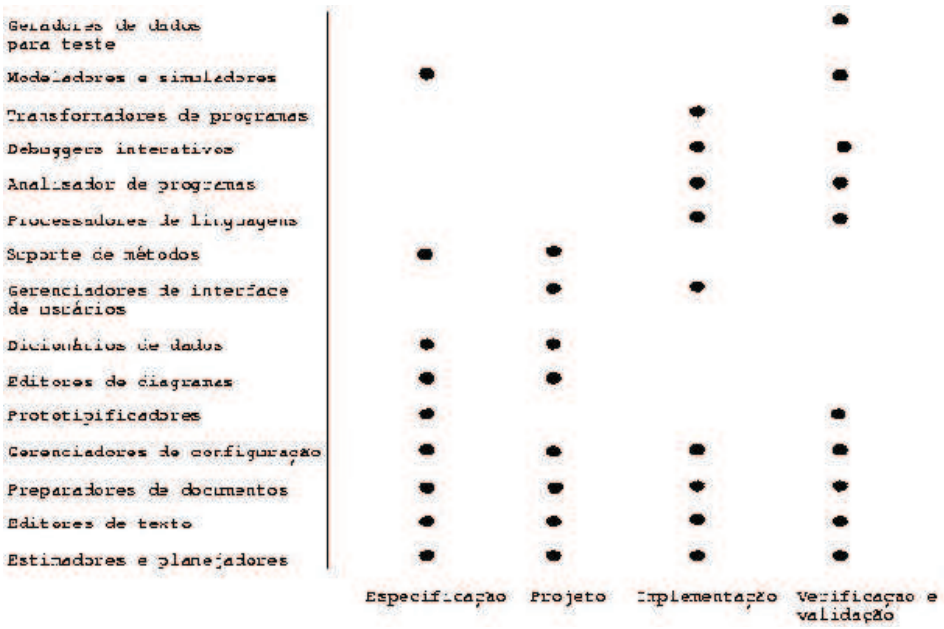

Figura 2.2 - Classificação de Ferramentas Vs. Atividades de Processo Suportadas

Os benefícios alcançados pelo uso de ferramentas são inúmeros, entre eles, destacam-se, aumento da capacidade de produção, automação de processos não-criativos, controle das atividades do processo e controle da capacidade de repetição de atividades iguais.

Algumas desvantagens no uso de ferramentas são provenientes de fatores como uso inadequado, seleção inadequada, forte dependência do fabricante, resistência às mudanças de processo provocadas pelo uso de ferramentas, dificuldade de integração das ferramentas e uso de ferramentas mal projetadas e mal desenvolvidas. Estas desvantagens provocam descrédito nos processos automáticos ou semi-automáticos e desperdício de tempo com atividades que seriam facilmente realizadas sem o auxílio de ferramentas.

2.3 Procedimentos

Os procedimentos garantem sincronismo e integração entre a utilização de ferramentas e os métodos adotados no processo de software. Em qualquer projeto de desenvolvimento é necessário selecionar ou definir o processo de software, tanto do ponto de vista técnico quanto gerencial. Do ponto de vista técnico, as mais importantes decisões e definições são a escolha das fases que irão compor o ciclo de vida do software e a escolha dos métodos que serão executados em cada fase.

As fases do ciclo de vida do software já estão praticamente consolidadas e sofrem poucas variações entre os projetos de desenvolvimento (análise de requisitos, especificação do sistema, projeto da arquitetura, projeto detalhado, implementação e integração, manutenção e evolução). Contudo, algumas aplicações específicas exigem fases diferenciadas e definidas visando as características particulares destas aplicações.

Devido ao aspecto técnico, os métodos a serem executados por fase exigem um esforço maior para serem definidos. Para a seleção dos métodos, são

considerados critérios que incluem maturidade, nível de suporte de ferramenta, tamanho da aplicação, formação do time de desenvolvimento, tipo de aplicação, entre outros.

Depois de formado o contexto, com os critérios citados, é escolhida a seqüência de métodos que irá suportar as fases do ciclo de vida do software, constituindo o que chamam de "modelo de processo de desenvolvimento de software".

A qualidade não é uma grandeza de fácil mensurabilidade, contudo, os modelos de processo de desenvolvimento fornecem uma representação abstrata para assegurar um mínimo de controle e garantia de qualidade do software a ser produzido [JALOTE, 1992; SOMMERVILLE, 1992].

Vários modelos de gerenciamento se consolidaram entre as empresas desenvolvedoras de software [HUMPHREY, 1997b]. Modelos que além de garantir os aspectos técnicos e gerenciais presentes nos modelos de processo, fornecem controle e garantia de qualidade do software. O modelo PSP pertence ao conjunto formado por estes modelos propostos para gerenciamento de software, contudo voltado à escala individual de gerência.

3 MODELOS DE GERÊNCIA DE PROCESSO

Neste capítulo serão apresentados os modelos de gerência de processo PSP e TSP.

Seguindo um modelo de gerenciamento de processo de software organizações têm alcançado melhorias significativas nos seus processos e modos de trabalho [KHOSHGOFTAAR et al, 1998; PHAM et al, 1999]. Contudo, muitas organizações perceberam que para obterem índices melhores dependiam, ainda, do talento individual de seus funcionários. Com isso, estudos e pesquisas foram direcionados para a capacitação e melhoria do processo pessoal. Processo pessoal são as técnicas, os métodos e as práticas utilizadas pelos recursos humanos de uma organização para executar tarefas.

O SEI, *The Software Engineering Institute*, com o objetivo de preencher a lacuna deixada pelos modelos de gerência de processo de software com relação ao processo pessoal, desenvolveu alguns modelos direcionados à melhoria de processo tanto de pessoal quanto individual. P-CMM [CURTIS et al, 1995], *People-Capability Maturity Model*, e PSP, *Personal Software Process* [HUMPHREY, 1997a], são os modelos apresentados pelo SEI como recursos para melhoria e otimização do processo pessoal e individual de trabalho, respectivamente.

Alguns modelos desenvolvidos sugerem práticas e métodos para que o próprio indivíduo consiga identificar e corrigir seus pontos fracos. Entretanto, o mérito do talento individual jamais será superado por tais técnicas. A criatividade e a capacidade de iniciativa necessária e presente nos bons funcionários não são cadastradas por tais modelos. Os modelos são sugestões para organizar e disciplinar os processos individuais e não diminuem nem restringem a capacidade criativa dos indivíduos.

3.1 O Modelo PSP – Descrição Geral

O PSP, *Personal Software Process,* foi desenvolvido para tornar o trabalho mais produtivo, adequado e satisfatório ao desenvolvimento de sistemas em escala individual, fazendo com que o próprio programador encontre seus limites.

Baseado no CMM [PAULK et al, 1993], *Capability Maturity Model* para software, o PSP também possui níveis e objetivos a serem alcançados a cada nível. A evolução do processo consiste em alcançar os objetivos para mudar de nível dentro do modelo. Estas evoluções são ilustradas na Figura 3.1. Como podem observar, a evolução é composta por fases que são seguidas até se alcançar o pleno controle sobre as atividades de desenvolvimento. As atividades de desenvolvimento são: planejamento baseado em dados obtidos com os relatórios, padrão de codificação, revisões de código e geração de relatórios de acompanhamento. Notam-se, também, os principais objetivos a serem alcançados em cada etapa do desenvolvimento.

As estratégias utilizadas na elaboração do PSP são listadas a seguir:

- Identificação de técnicas e métodos utilizados em sistemas de grande escala que possam ser úteis para os sistemas individuais;
- Definição de um subconjunto destes métodos e técnicas para serem aplicados no desenvolvimento de pequenos programas;
- Estruturação destes métodos para que sejam gradualmente introduzidos ao modelo PSP;
- Fornecimento de um conjunto de exercícios a serem realizados, possibilitando o aprendizado do PSP.

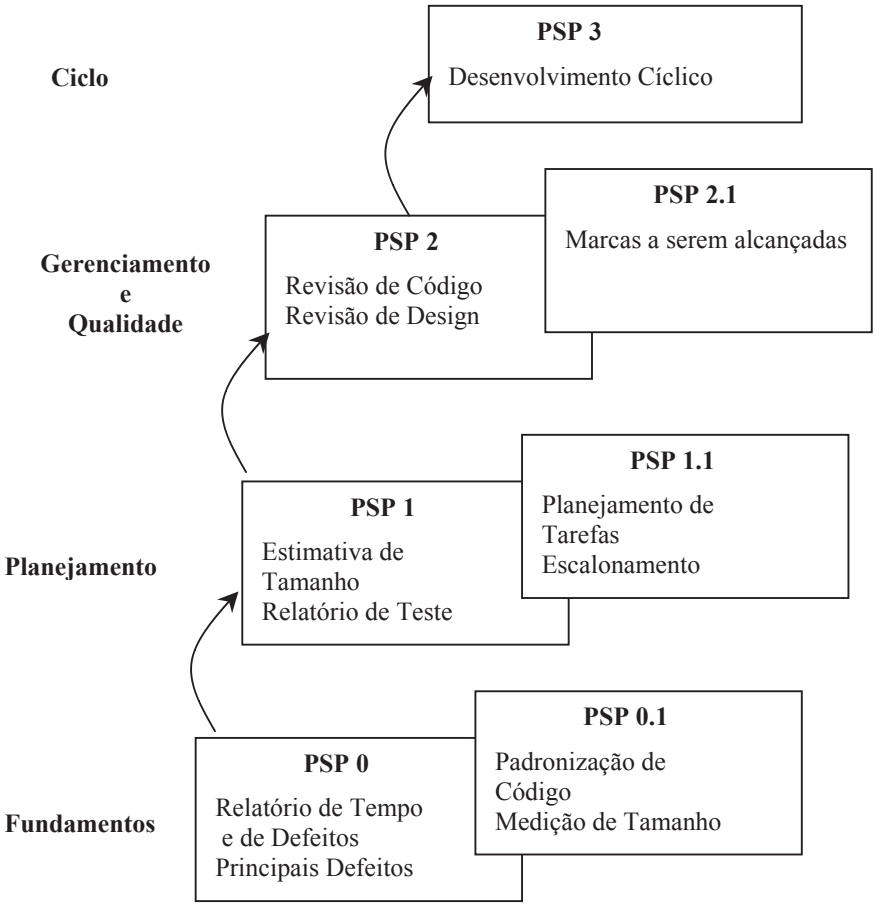

Figura 3.1 - Evolução do PSP

3.2 O Modelo TSP – Descrição Geral

Grande parte dos softwares é desenvolvida por equipes. Para produzir produtos de qualidade e cumprir planejamentos audaciosos, o trabalho em equipe é fundamental. Para que uma equipe alcance seus objetivos, cada membro desta deve empenhar-se e dedicar-se o máximo possível. Os membros devem estar cientes que o objetivo da equipe é o objetivo de cada membro. Mas

formar uma equipe não é uma simples tarefa. Podem surgir problemas que venham a causar falhas no sucesso da equipe.

O SEI, *The Software Engineering Institute,* com o objetivo de suprir a falta de modelos de processos voltados a equipes de desenvolvimento, criou o TSP, *Team Software Process*. O TSP é uma evolução do PSP visando a qualidade do processo em equipes de desenvolvimento [HUMPHREY, 1999]. Este modelo de processo guia a equipe através de passos e ciclos, como no PSP.

O TSP é baseado em quatro princípios básicos:

1. O aprendizado é mais efetivo quando se segue um processo definido e consegue-se um rápido retorno. Através dos formulários e roteiros do TSP, tem-se um processo definido e calculado para equipes de engenharia de software. Como no modelo TSP a equipe desenvolve o produto em vários ciclos e no final de cada ciclo é realizada uma avaliação do produto desenvolvido, tem-se assim um rápido retorno;
2. Um trabalho produtivo de equipe, requer a combinação de objetivos específicos e liderança capacitada. No TSP, o objetivo do projeto é construir um produto e um membro da equipe será o líder;
3. Quando se batalha com atuais problemas no projeto e tomam-se soluções efetivas para resolvê-los, deseja-se que os benefícios sejam vistos. Sem utilizar o TSP, pode-se perder muito tempo definindo práticas e métodos próprios;
4. A instrução é mais efetiva quando é construída em conhecimentos prévios. O TSP foi construído baseado na experiência com equipes de software e cursos de equipes de software.

O TSP utiliza-se de uma estratégia de desenvolvimento cíclico para construir um produto final, como ilustrado na Figura 3.2. Apesar de que esta figura não apresenta um modelo rígido. A equipe de desenvolvimento poderá escolher em quantos ciclos poderá desenvolver o sistema. Conforme

[HUMPHREY, 1999], quanto maior o número de ciclos maior será o controle e desenvolvimento de um produto de qualidade. As fases que compõem um ciclo também podem sofrer modificações conforme as necessidades da equipe. De acordo com a Figura 3.2, o primeiro ciclo inicia-se com o lançamento do projeto em aspectos gerais. A partir deste lançamento, a equipe segue o ciclo através de sete fases:

- Estratégia;
- Planejamento;
- Requisitos;
- Projeto;
- Implementação;
- Testes e;
- Avaliação.

Nos ciclos seguintes, a equipe segue os mesmos passos, mas tendo como base parte do projeto desenvolvido nos ciclos anteriores. Se algum problema ocorrer durante o ciclo, soluções podem ser propostas para serem aplicadas nos ciclos posteriores.

A estratégia de desenvolvimento cíclica utilizada pelo TSP, parte do princípio "Dividir para Conquistar". O produto final é dividido em partes menores que serão construídas durante os ciclos de desenvolvimento estabelecidos pela equipe. Ao término de cada ciclo, pode-se, com os resultados dos ciclos anteriores, estimar o tamanho e tempo de desenvolvimento para a parte do produto que será desenvolvida no próximo ciclo. Utilizando-se desta estratégia deve-se considerar que:

1- Em cada ciclo deve ser produzida uma versão testável que será parte do produto final;
2- Cada ciclo deve ser pequeno embora deve ter tempo suficiente, para desenvolvimento e testes das versões a serem produzidas;

3- Quando as versões desenvolvidas durante os ciclos forem agrupadas, estas deverão produzir o produto final desejado.

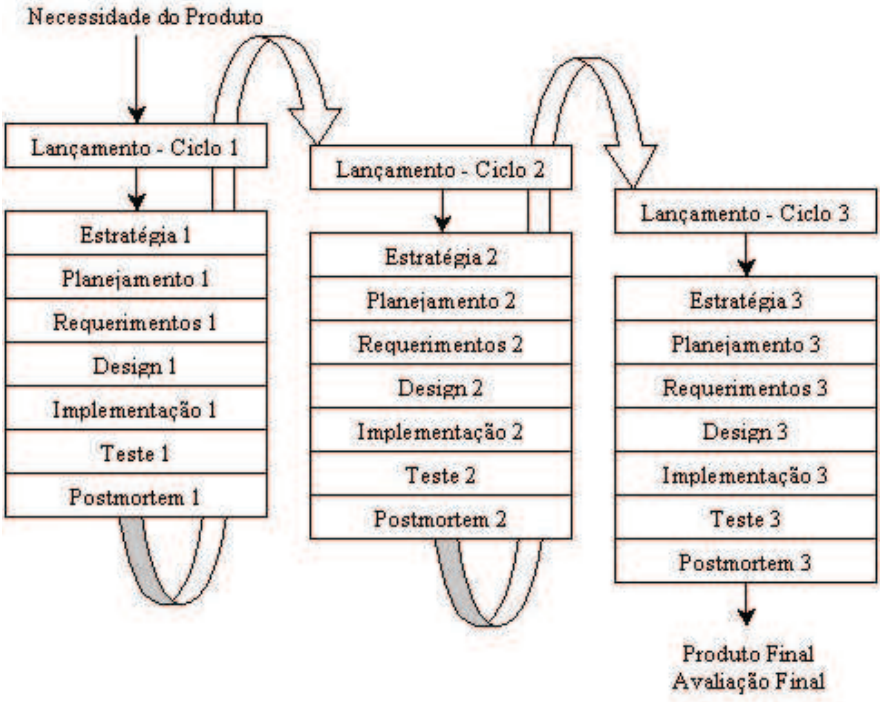

Figura 3.2 - Estrutura e Fluxo TSP em um projeto de 3 ciclos

3.2.1 As fases que compõem o TSP

Cada ciclo de desenvolvimento do TSP é composto por oito fases, mas este número pode variar dependendo da necessidade da equipe. Estas fases são lançamento, desenvolvimento de estratégia, planejamento, requisitos, projeto, implementação, testes e avaliação.

3.2.1.1 Lançamento

Na fase de lançamento são definidos os seguintes tópicos:

- quem serão os membros da equipe;
- qual o papel de cada membro na equipe;
- o objetivo da equipe;
- o produto que se deseja produzir.

Os papéis que cada um desempenha dentro da equipe podem mudar nos ciclos seguintes visto que algum membro pode não se identificar com o papel que lhe foi designado. São definidas reuniões semanais para se observar o andamento do projeto e o desempenho da equipe.

3.2.1.2 Estratégia

Tendo em vista que toda a equipe está formada e ciente do projeto, nesta fase a equipe desenvolve a estratégia para realizar os trabalhos durante o ciclo. Será criado um modelo conceitual, estimativas de tamanho e tempo de desenvolvimento do produto. Se o tempo ultrapassar o que foi previsto para o ciclo, a estratégia deverá ser revista e um novo módulo funcional do produto deverá ser definido para ser desenvolvido durante o ciclo. A fase de estratégia é realizada antes que o planejamento, pois nesta fase serão gerados artefatos que serão úteis na fase de planejamento. Nesta fase a equipe começa a produzir o plano de gerência de configuração. Este plano é fundamental para o controle da versão do produto. Quando uma equipe desenvolve um produto sempre há problemas de coordenação e o plano de gerência de configuração resolve este problema.

3.2.1.3 Planejamento

Há várias razões para se fazer um plano. Quando se tem um plano detalhado, sabe-se com exatidão o que deve ser feito e quando será feito.

Quando não se usa o planejamento, a equipe leva muito tempo para desenvolver o projeto. O plano pode variar dependendo do contexto em que é aplicável. O planejamento ainda é uma fase que requer tempo e paciência. Entretanto, como o TSP utiliza-se da estratégia de desenvolvimento cíclico e são desenvolvidas pequenas versões, os planos são simples [HUMPHREY, 1999].

Um dos problemas do cumprimento de tarefas é o trabalho não balanceado [HUMPHREY, 1999]. Em uma equipe sincronizada, todos os membros terminam suas tarefas em uma ordem correta e no mesmo tempo. No TSP, a equipe desenvolve o planejamento, distribuindo as tarefas de forma balanceada. Este balanceamento sempre é verificado nas fases de avaliação dos ciclos de desenvolvimento.

Na fase de estratégia, gerou-se o modelo conceitual do projeto. É a partir deste modelo que o plano será confeccionado. Serão determinados artefatos relacionados como especificação de requisitos, planos de testes. Todos os produtos desenvolvidos neste ciclo deverão ser listados e estimados seus tamanhos. Com base nestes dados, também será criado o plano de qualidade.

O plano de qualidade prevê medidas como tempo de desenvolvimento, número de erros por KLOC (mil linhas de código), porcentagem livre de erros, porcentagem de reusabilidade, porcentagem de remoção de erros etc. Nos finais dos ciclos, estas medidas são utilizadas para uma avaliação do controle de qualidade do produto a ser produzido.

3.2.1.4 Requisitos

Na fase de requisitos, a equipe desenvolve a especificação de requisitos do software. Este artefato prevê exatamente o que o produto será. Através dele é que será feita a avaliação final do produto. Será constatado se o produto final é realmente o que o cliente desejava. Uma das causas da insatisfação do cliente é a

má funcionalidade do software. Criando a especificação, a equipe evita este problema. Para gerar esta especificação, relatam-se todas as necessidades do cliente em forma de perguntas. A partir do momento que todas as perguntas estiverem respondidas, tem-se realmente o que o cliente deseja.

Existem muitos padrões para documento de requisitos, entretanto o TSP se concentra nos requisitos funcionais e operacionais. Entre eles estão:

- Requisitos funcionais: entradas, saídas, cálculos e casos de uso;
- Requisitos de Interface Externa: usuário, hardware, software, comunicações;
- Restrições de Projeto: formato de arquivos, linguagens, padrões, compatibilidade e assim por diante;
- Requisitos Não-Funcionais: segurança, conversão, rastreabilidade, usabilidade e assim por diante;
- Outros requisitos: banco de dados, instalação etc.

No TSP, o objetivo principal do documento de requisitos é documentar os acordos da equipe em relação às funcionalidades e interfaces externas.

3.2.1.5 Projeto

O projeto trata-se de uma fase onde a equipe irá decidir como construir o produto. Não somente descrever idéias gerais, mas produzir uma especificação completa e precisa de como o produto será construído.

Quando um projeto é mal definido, os membros da equipe perdem tempo resolvendo problemas da má especificação. Esta situação gera incompatibilidades entre os componentes desenvolvidos criando um produto de má qualidade. Ao contrário, quando um projeto é bem definido, o produto gerado segue exatamente o que foi descrito no documento de requisitos, proporcionando qualidade, usabilidade e a satisfação do cliente.

No TSP, a construção do projeto consiste dos seguintes passos:

- Decidir a estrutura do produto como um todo;
- Alocar as funcionalidades do produto em componentes;
- Produzir a especificação externa dos componentes e;
- Decidir quais os componentes e as funcionalidades que serão desenvolvidas em cada ciclo.

3.2.1.6 Implementação

A fase de implementação é a geração do código fonte do produto. Esta implementação deve seguir o que foi proposto na fase de projeto. Os principais passos na fase de implementação são: planejamento de implementação, projeto detalhado, inspeção do projeto detalhado, codificação, inspeção de código, unidade de testes, revisão da qualidade dos componentes e lançamento dos componentes.

Uma das características da fase de implementação no TSP é a utilização de padrões. No início, pode-se perder um certo tempo definindo padrões de implementação. Mas este tempo perdido será muito bem recompensado adiante com o uso dos padrões. Os padrões de implementação utilizados são:

- padrões de revisão
- padrões de nomeação, interface e mensagens
- padrões de codificação
- padrões de tamanho
- padrões de defeitos
- prevenção de defeitos.

3.2.1.7 Integração e Testes

Nesta fase, os módulos desenvolvidos no ciclo são testados e integrados para formar o produto final. O objetivo é verificar se tais módulos produzirão um produto de qualidade. As principais atividades de testes no TSP são:

- Utilizar as partes desenvolvidas e testadas para construir o produto final;
- Utilizar a integração e testes no produto, para verificar se este está apropriadamente construído, que todos os módulos estão presentes e que funcionam perfeitamente juntos;
- Testar o produto para avaliar se ele atende aos requisitos.

Durante estas atividades, também é desejado:

- Identificar módulos com baixa qualidade e enviá-los para o gerente de qualidade/processo para revisão e correção;
- Identificar módulos que já foram enviados para o gerente de qualidade/processo, que ainda possuem baixa qualidade e reenviá-los novamente para verificação e correção.

De um modo geral a fase de Testes é composta pela construção, testes de integração e teste do produto. A construção é caracterizada pela união dos módulos para compor o produto final. Os testes de integração garantem que todos os módulos estão presentes e que trabalhem perfeitamente juntos. O teste do produto avalia as funcionalidades e desempenho em relação aos requisitos especificados pelo cliente.

A documentação é uma parte essencial de um software de qualidade. Em alguns casos, ela é mais importante que o código do programa [HUMPHREY, 1999]. É na fase de testes que a documentação é criada e revista. Enquanto alguns engenheiros estão ocupados com testes, outros devem ficar responsáveis pela documentação do produto.

3.2.1.8 Análise

A análise é a fase final do TSP. Nela todo o trabalho da equipe é revisto para garantir que tudo o que foi planejado foi realmente cumprido. Também é verificado se todos os dados foram registrados pelos membros da equipe. É analisado o que foi desenvolvido no ciclo e quais são as melhorias para o próximo ciclo ou para o próximo produto. Se o modo de como o trabalho é feito não for mudado, os resultados adquiridos serão os mesmos.

Através da análise, serão definidas melhorias tanto no processo de desenvolvimento quanto no processo gerencial da equipe. No aspecto gerencial serão analisados os rendimentos, desempenho, pontos de vista e dificuldades dos engenheiros que desempenharam os papéis de gerentes e líder. Com os resultados em mãos, verificar a necessidade de mudar os membros que desempenharam tais papéis. Toda a análise tem como objetivo prover um método par avaliar todo o processo de desenvolvimento e estipular melhorias

O TSP já vem sendo implantado em algumas empresas e apresentado resultados satisfatórios [GOTH, 2000]. Este modelo, em relação à Engenharia de Software, atende aos requisitos "**métodos**" e "**procedimentos**" porém falha no requisito "**ferramenta**". Ainda faltam ferramentas adequadas que auxiliem na utilização deste [GOTH, 2000]. Atualmente já se encontram disponíveis somente planilhas EXCEL que auxiliam no preenchimento dos roteiros, mas nada ainda que fosse padrão. Vislumbrando esta problemática, este trabalho visa propor uma arquitetura de software que dê apoio à utilização do modelo TSP.

4 O *PROJECTSPACE* E O TSP

De forma distinta dos processos de manufatura, que podem ser definidos e automatizados em operações de linha de produção, os processos de software são compostos por atividades que são realizadas por humanos e são atividades intelectuais e criativas envolvendo muita colaboração [GIBBS, 1994]. A engenharia de processo tem se esforçado no sentido de definir modelos e padrões para a construção de um efetivo processo de desenvolvimento de software para as organizaçoes tais como [CROMER et al, 1999; GATES, 1997; MAIDANTCHIK et al, 1999]. O reconhecimento da importância dos processos e o crescimento da cultura de processo têm levado as organizações à criação de ambientes de engenharia de software mais eficientes. Um exemplo são os PSEEs - *Process-Centered Software Engineering Environment*. Os PSEEs integram tanto os requisitos do produto, que são o foco da engenharia de software, como os requisitos do processo, que são o foco do gerenciamento do projeto e da engenharia do processo.

Todavia, a maioria das organizações de software ainda sentem dificuldade em definir os seus processos padrões, muitas vezes por nem sequer saberem gerenciar os seus projetos de forma adequada [FERNANDES et al, 1989; MAIDANTCHIK et al, 1999]. O gerenciamento de projeto tem como objetivos, entre outros, assegurar que processos particulares sejam seguidos, coordenando e monitorando as atividades da engenharia de software. Porém, o gerenciamento de projetos de software ainda é pouco abordado e praticado. Os ambientes tradicionais de engenharia de software têm suportado somente a engenharia de software, assumindo um processo implícito e tendo como foco principal o produto. Esta visão limita as organizações no que diz respeito à tomada de decisões, ao estabelecimento e arquivamento de metas organizacionais, à determinação de pontos para melhoria, à estipulação de

prazos para entrega de produtos, à obtenção de uma certificação, entre outros. Além dessas limitações, os ambientes de gerenciamento de projetos disponíveis no mercado não são específicos para software impossibilitando a obtenção de métricas automáticas, sem exigir grande intervenção humana.

Este capítulo descreve a arquitetura *ProjectSpace* e sua utilização na implantação do modelo TSP.

4.1 Arquitetura do *ProjectSpace*

O *ProjectSpace* é composto por oito subsistemas principais que cooperam entre si, como ilustra a Figura 4.1:

- *TimeSheet* – fornece apoio no processo de disciplina pessoal realizando o registro e avaliação do tempo gasto com cada atividade;
- *ResourceManager* – Gerencia os recursos da organização tais como: espaço físico, pessoal, treinamentos, ferramentas, habilidades e funções individuais e de equipes, etc. Além de proporcionar uma previsão da liberação destes recursos;
- *RequirementManager* – Gerencia os requisitos dos sistemas de software de cada projeto efetuando o controle dos requisitos por sistema de software, associando os requisitos às aplicações (ou atividades) a serem desenvolvidas, promove uma visão da realização destes requisitos;
- *ProcessManager* – Gerencia os elementos do processo de software como: atividades, métodos, técnicas, modelos de ciclo de vida, padrões e normas de qualidade, classe de problemas do desenvolvimento, etc. Tem por função também auxiliar a confecção de planos padrão de projetos e dar uma visão de problemas gerais do processo de software;
- *QualityManager* – Gerencia os atributos de qualidade do processo de software efetuando o cadastramento de modelos e normas e mapeando a dimensão de processo para as atividades da organização. Deve propiciar

uma avaliação parcial do processo de software, no que se refere à gerência de projetos;

- *PlanEngine* – Gerencia os processos de desenvolvimento de software instanciados para um projeto específico. Deve ter funções como: auxílio na confecção do plano de projetos, acompanhamento da execução do plano, registro das versões e alterações do plano, registro das versões e alterações das atividades dos planos, visualização do andamento dos trabalhos, ativação das ferramentas CASE através do ToolManager, comunicação parcialmente automatizada, visualização das restrições do projeto e das atividades, registro de tempo gasto nas atividades de desenvolvimento, etc.;
- *ToolManager* – Gerencia e ativa as ferramentas CASE através do registro e controle das ferramentas CASE;
- *DataAccessLayer* – Realiza o acesso às informações armazenadas na base compartilhada por todos os subsistemas do *ProjectSpace*.

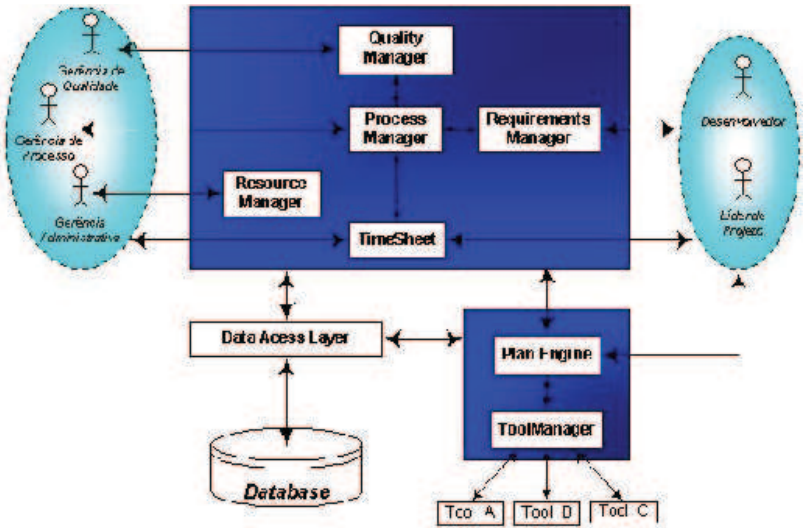

Figura 4.1 – Arquitetura do *ProjectSpace*

4.2 Metodologia Utilizada para a Construção do *ProjectSpace*

Inicialmente foi realizado um levantamento dos requisitos gerais do *ProjectSpace* através do estudo de padrões e normas de qualidade, metodologias de desenvolvimento de software, modelos de ciclo de vida, processos de software, ambientes de desenvolvimento, ferramentas CASE, métricas de projeto, gerenciamento prático de projetos de software, entre outros. Após o levantamento dos requisitos uma especificação conceitual do *ProjectSpace* foi feita. Esta especificação vem sendo constantemente modificada conforme os módulos são implementados e validados junto ao CESAR e a Devex.

A construção dos módulos foi iniciada adotando uma metodologia baseada nas seguintes etapas:

1 Levantamento dos requisitos do módulo;
2 Modelagem das informações e criação dos modelos conceituais mais específicos;
3 Especificação e construção do módulo;
4 Implementação do módulo, como protótipo, de forma diferenciada no CESAR e na Devex;
5 Avaliação do uso destes módulos junto aos projetos de software na Devex e no CESAR e levantamento das métricas para melhoria do processo de software;
6 Nova especificação e implementação dos módulos considerando os problemas detectados e melhorias necessárias;
7 Validação e avaliação da nova arquitetura;
8 Integração no projeto como um todo.

Inicialmente foram escolhidos para serem implementados e avaliados os módulos *TimeSheet* e *PlanEngine* por serem considerados os mais críticos dentro do projeto. Desta forma, as etapas um (1) a cinco (5) para estes dois

módulos da metodologia foram realizadas no CESAR e na Devex. O trabalho aqui apresentado foi feito no protótipo construído na Devex e representa uma das partes da etapa cinco (5) da metodologia aqui apresentada. O *PlanEngine* da Devex que foi implementado na plataforma *Lotus-Notes*. Escolheu-se a utilização desta plataforma em função do desenvolvimento de software ser um trabalho que envolve muita colaboração humana e devido à experiência da empresa em desenvolvimento de sistemas nesta plataforma. Um maior detalhamento deste protótipo pode ser encontrado em [ROUILLER et al, 2001b]. Os demais módulos do *ProjectSpace* se encontram em fase de implementação do protótipo, referente a etapa quatro (4) da metodologia.

4.3 O *ProjectSpace* frente ao Modelo TSP

Pode-se verificar em [ROUILLER et al, 2001a] que alguns dos subsistemas do *ProjectSpace* já vêm sendo utilizados na empresa Devex Tecnologia e Sistemas. Os subsistemas se encontram em fase de avaliação e já gerenciam parcialmente os processos de desenvolvimento de software desta empresa. Tendo em vista esta utilização, a arquitetura vista na seção anterior e a necessidade de ferramentas que auxilie na implantação do TSP pode-se propor a utilização do TSP com o ProjectSpace, como ilustrado na Figura 4.2.

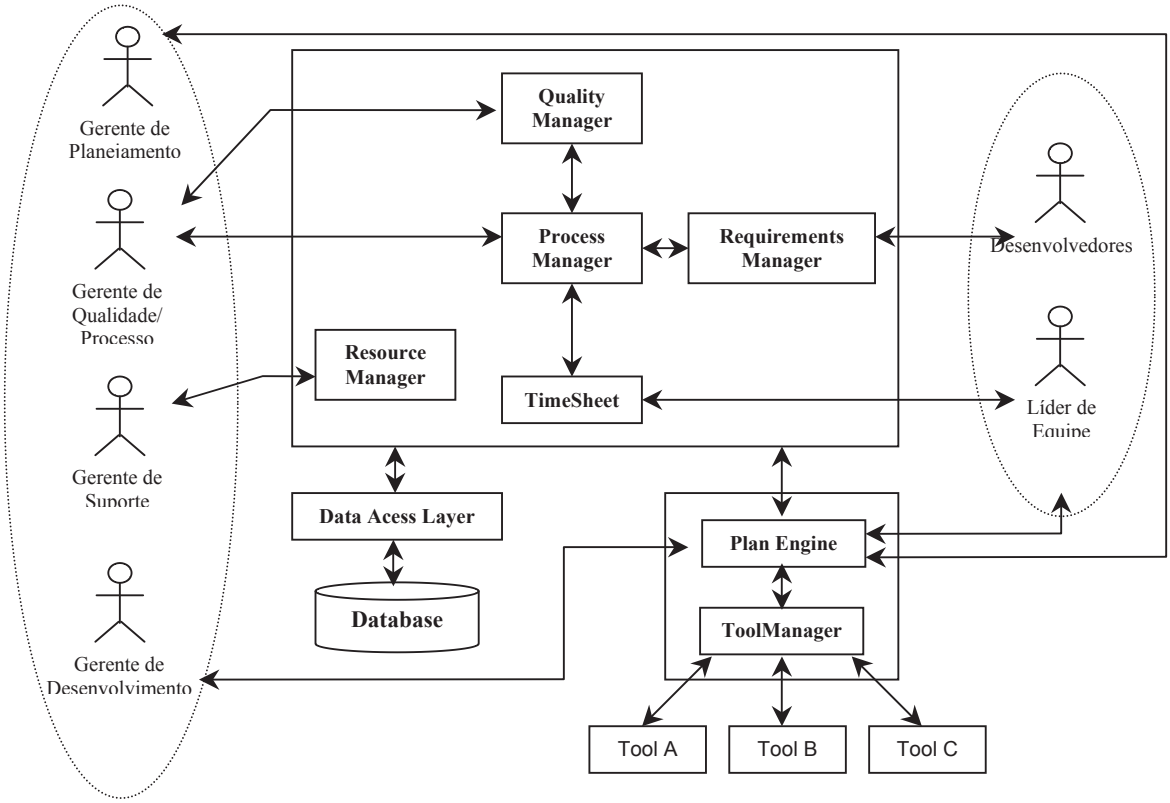

Figura 4.2 - Arquitetura do *ProjectSpace* aplicada ao TSP

Observa-se, pelas funcionalidades descritas na seção anterior, que o *ProjectSpace* fornece o apoio necessário para a implantação do TSP. Abaixo segue as facilidades que cada módulo oferece para os papéis que os membros das equipes realizam durante um processo de desenvolvimento de software:

- *TimeSheet* – através do registro do tempo das atividades desenvolvidas, o *TimeSheet* auxilia no preenchimento dos roteiros responsáveis pelo controle das atividades que os engenheiros desenvolvem durante a semana ou ciclo. Estes roteiros são essenciais para a avaliação da qualidade da equipe e do desenvolvimento do projeto;

- *ResourceManager* – O gerente de suporte, acessando este módulo, obtém informações sobre as necessidades que a equipe possui no desenvolvimento do projeto, como ferramentas, espaço físico, treinamentos, etc;
- *RequirementManager* – Este módulo auxilia os desenvolvedores na construção do software fornecendo os requisitos do software em questão. Tem fundamental importância na fase de Requisitos do TSP;
- *ProcessManager* – Auxilia o gerente de qualidade/processo no controle do modelo de desenvolvimento cíclico adotado pelo TSP, tendo acesso às atividades do software a ser desenvolvido, normas de qualidade e problemas do desenvolvimento;
- *QualityManager* – Juntamente com o *ProcessManager*, auxilia o gerente de qualidade/processo no controle dos atributos de qualidade do software. Tem acesso às informações de erros/KLOC, porcentagem de correção de erros/KLOC, etc;
- *PlanEngine* – Auxilia o gerente de planejamento e a equipe na confecção dos planos de projetos do software. Através do acompanhamento da execução do plano, dá informações para que o gerente de planejamento produza um plano de tarefas para os ciclos seguintes. Auxilia no controle de versão do software;
- *ToolManager* – Pode manter comunicação com compiladores obtendo informações, como erros de sintaxe, que auxiliam no controle da qualidade do software;

4.4 Validando o *PlanEngine* e o *TimeSheet* da Devex

Esta seção aborda as funcionalidades do *PlanEngine* e do *TimeSheet* da Devex comparando com as reais necessidades impostas pelo modelo de gerência TSP.

A Figura 4.3 ilustra uma visão das horas gastas por colaborador em um determinado projeto. Um dos pontos negativos observados no protótipo é que as atividades cadastradas são somente as realizadas e não as planejadas. No final das atividades, o executor cadastra a atividade e informa a quantidade de horas gastas. A falta do cadastro de horas planejadas tem um impacto na avaliação da qualidade da equipe que está desenvolvendo o projeto. Outro ponto negativo observado é que o protótipo está adaptado ao ciclo de vida de software adotado pela Devex. Para adaptá-lo ao modelo TSP, melhorias deverão ser feitas no protótipo. Tais melhorias englobam a possibilidade de realizar o cadastro de qualquer modelo de ciclo de vida.

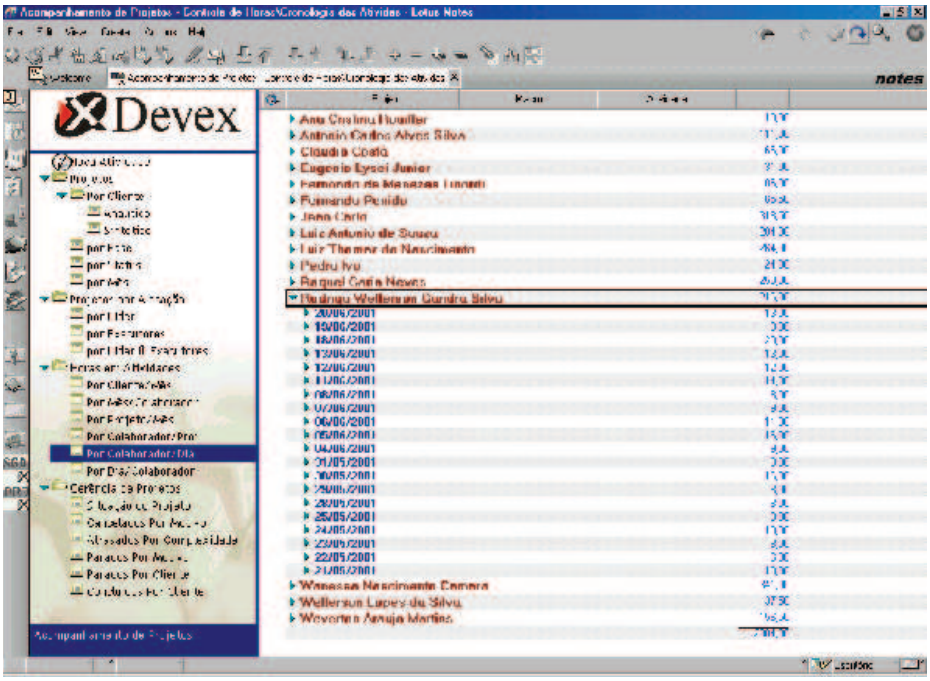

Figura 4.3 - Uma visão das horas gastas por colaborador no PlanEngine

A Figura 4.4 ilustra o cadastro de atividades. Como pontos positivos nesta visão pode-se citar a possibilidade de se avisar ao líder da equipe, por e-mail, sobre o cadastramento da atividade. Através deste e-mail, o líder tem condições de verificar problemas na execução da atividade, verificar observações, horas gastas.

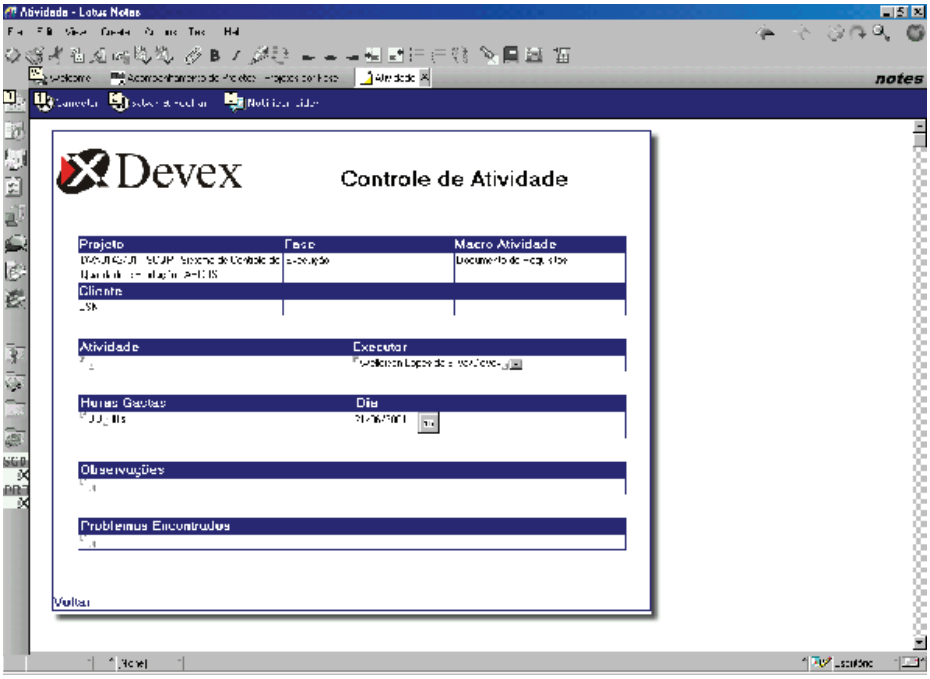

Figura 4.4 - Visão do cadastramento de atividades

A Figura 4.5 ilustra as informações sobre uma determinada fase do projeto. Nela são exibidas informações importantes para a análise do processo, como o total de horas previsto da fase. Outra funcionalidade importante é a possibilidade de marcar uma reunião com os membros da equipe. Esta funcionalidade atende à necessidade da tarefa do líder da equipe de marcar reuniões semanais para discutir o andamento do projeto. Todos os membros da equipe recebem um e-mail notificando a reunião.

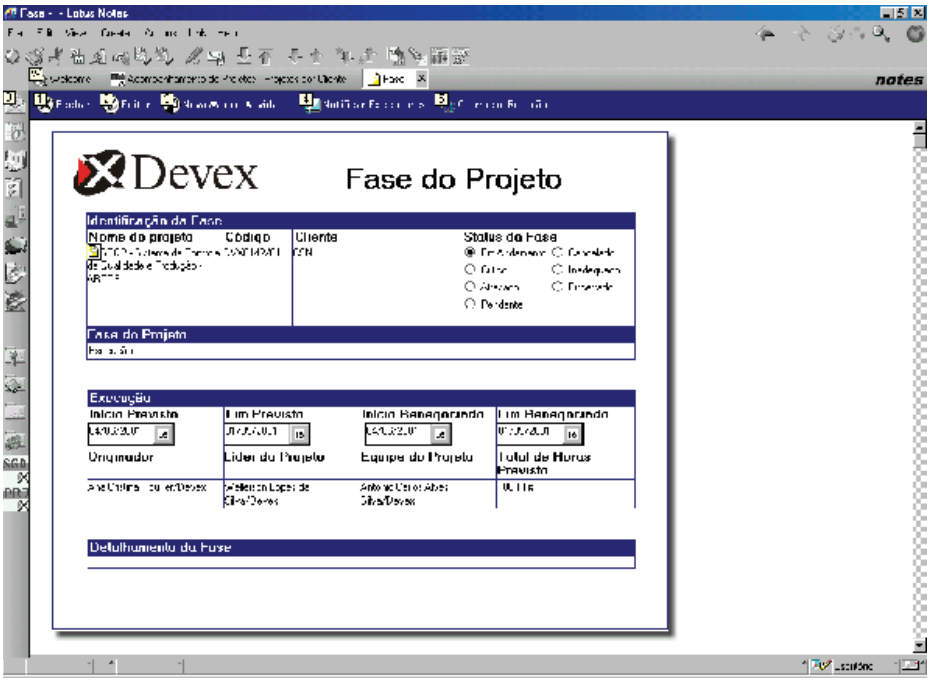

Figura 4.5 - Visão das informações de uma fase do projeto

5 CONCLUSÃO E FUTUROS TRABALHOS

Avaliando e analisando a arquitetura do *ProjectSpace* e utilização do protótipo na Devex, percebe-se que é possível a utilização do *ProjectSpace* em conjunto com o TSP. Apesar, que para utilizar o protótipo da Devex devem ser realizadas modificações como as vistas no Capítulo 4.

Observam-se como pontos positivos da implantação do protótipo na Devex:

- Maior e melhor controle do esforço despendido pela equipe técnica nas diversas fases do projeto;
- Melhor relação da quantidade de horas especificadas e quantidade de horas realizadas;
- Qual o esforço despendido em cada fase e em relação uma a outras;
- Registro e melhor formalização de todos os projetos da empresa;
- Melhor notificação de problemas encontrados em campo para os níveis mais altos das organizações tais como: gerencias e diretorias;
- Notificação ativa de mudança de fase nos projetos;
- Notificação aos executores das macro-atividades registradas;
- Maior registro de todas as atividades realizadas;
- Melhor confecção do plano de projeto por demanda;
- Maior e melhor acompanhamento das diversas fases do projeto;
- Maior registro dos problemas ocorridos durante a execução dos projetos;
- Melhor visualização do andamento dos trabalhos;
- Melhor visualização dos pré-requisitos de atividades e projetos;
- Maior registro do tempo gasto nas atividades;
- Melhor avaliação parcial do processo de software.

Alguns problemas também foram detectados tanto na implantação do sistema de gerência de projetos como em sua utilização, tais como:

- Dificuldade para o aculturamento do apontamento de horas pelas pessoas da organização;
- Necessidade de tipificação das macro-atividades para obtenção de métricas mais gerais;
- Necessidade do registro histórico das alterações/correções nos diversos artefatos e atributos dos artefatos do produto;
- A realização da confecção dos planos por demanda em quase todos os projetos;
- Falta de avaliação de riscos.

Como trabalhos futuros pretende-se trabalhar ativamente no projeto da nova ferramenta, em construção, com o intuito de se atender as necessidades da Devex e do modelo de gerência TSP.

Como melhorias, facilidades e resultados futuros de uma implementação bem sucedida pode-se ter entre outros: o escopo do trabalho do projeto definido; a viabilidade de alcançar as metas do projeto baseada na verificação dos recursos disponíveis e restrições; as tarefas e recursos necessários para completar o trabalho medidas e estimadas; as interfaces entre os elementos do projeto, e com outros projetos e unidades organizacionais identificadas e monitoradas; os planos para execução do projeto desenvolvido e implementado; o progresso do projeto monitorado e registrado; as ações para corrigir os desvios do plano e prevenir a recorrência de problemas identificados no projeto; etc.

Algumas métricas também são esperadas tais como: Quão bem distribuídos estão os recursos da organização (jornada de trabalho, espaço físico, licenças de ferramentas CASE, etc.); Quais as habilidades e funções mais comuns na organização; Qual a necessidade de um determinado treinamento; Quais as metodologias, os modelos de ciclo de vida, técnicas, recursos,

ferramentas, etc. mais utilizados na organização; Quão maduro um indivíduo ou uma equipe está para utilizar determinada tecnologia; Quais os problemas que mais ocasionam atrasos em projetos; Em que fases há uma maior incidência de problemas nos projetos; Quais os problemas que mais levam a adaptação ou confecção de um novo plano de projeto; Quão bem é estipulado o término das atividades dos projetos; Quanto se realiza as atividades dentro dos prazos estabelecidos; Quanto se cumpre o que foi planejado; Quanto o projeto já evoluiu; Quais os problemas que mais ocorrem no desenvolvimento de sistemas de software; Quanto se cumprem as práticas básicas das normas em um dado projeto ou no desenvolvimento como um todo; etc.

6 REFERÊNCIAS BIBLIOGRÁFICAS

1. BOEHM, W.B. **Software Engineering Economics**. Prentice-Hall, 1981.

2. CROMER, T.; HORCH, J. **From the many to the one-one companys path to standardization**. IEEE, 1999.

3. CURTIS, B.; HEFLEY, W.E.; MILLER, S. et al. **The people-cmm**. SPN - Software Process Newsletter, 4, Fall 1995.

4. FERNANDES, A.A.; KUGLER, J.L.C. **Gerência de Projetos de Sistemas**. Rio de Janeiro, LTC, 1989.

5. FLECHER, T; HUNT, J. **Software Engineering and Case, Bridging the Culture Gap**. McGraw-Hill, 1993.

6. GATES, L.P. **How to Use the Software Process Framework**. Special report CMU/SEI-97-SR-009, 1997.

7. GIBBS, W. **Software's chronic crisis**. Scientific American, September, 1994.

8. GOTH, G. **The Team Software Process: A Quiet Quality Revolution**. IEEE Software, Nov-Dec 2000.

9. HUMPHREY, W.S. **Introduction to the Personal Software Process**. Adison Wesley, 1997a.

10. HUMPHREY, W.S. **Introduction to the Team Software Process**. Adison Wesley, 1999.

11. HUMPHREY, W.S. **Results os applying the personal software process**. IEEE Computer, p.24-31, 1997b.

12. JALOTE, P. **An Integrated Approach to Software Engineering**. Narosa Publishing House, 1992.

13. KARLSSON, J; RYAN, K. **A cost-value approach for prioritizing requirements**. IEEE Software, p.67-74, Sep-Oct 1997.

14. KHOSHGOFTAAR, T.M. et al. **Using process history to predict software quality**. IEEE Computer, p.66-72, Apr 1998.

15. KITCHENHAM, B; TAYLOR, N.R. **Software project development cost estimation**. Journal of Systems and Software, v.5, p.267-278, 1985.

16. MOORE, L.J; CLAYTON, E.R. **Gert Modelling and Simulation: Fundamentals and Applications**. Petrocelli/Chapter, 1976.

17. MAIDANTCHIK, C.; ROCHA, R.C.; XEXEO, G.B. **Software Process Standardization for Distributed Working Groups**. In Proceedings of the 4 th IEEE International Software Engineering Standards Symposioum, Curitiba, Paraná, Brasil, Mai 1999.

18. PAULK, M.; CURTIS, B.; CHRISSIS, M. et al. **Capability maturity model for software, version 1.1**. Technical Report CMU/SEI-93-TR24, SEI - Software Engineering Institute, 1993.

19. PHAM, H.; ZHANG, X. **A software cost model with warranty and risk costs**. IEEE Transactions on Computers, v.48, cap.1, p.71-75, Jan 1999.

20. PILLAI, K; NAIR, V.S.S. **A model for software development effort and cost estimation**. IEEE Transactions on Software Engineering, v.23, cap.8, p.485-497, Aug 1997.

21. PRESSMAN, R.S. **Engenharia de Software**. Makron Books, 1995.

22. ROUILLER, A.C.; MEIRA, S.R.L.; ALVARENGA, G.B. et al. **Gerenciamento Automatizado de Projetos de Software: Uma Experiência Prática na Devex Tecnologia**. XII Conferência Internacional de Tecnologia de Software. Jun 2001a.

23. ROUILLER, A.C.; MEIRA, S.R.L.; ALVARENGA, G.B.; et al. ***PlanEngine*: Ferramenta para Gerenciamento Automatizado de Projetos de Software**. XII Conferência Internacional de Tecnologia de Software. Jun 2001b.

24. SOMMERVILLE, I. **Software Engineering**. Addison Wesley, 1992.

25. ZARELLA, P.F. **Case tool integration and standardization**. Technical Report CMU/SEI-90-TR-14, Software Engineering Institute, 1990.