**UM SISTEMA DE GERÊNCIA DE PROJETOS DE
DESENVOLVIMENTO DE SOFTWARE PARA A
ESTAÇÃO TABA**

Elisiane Amaral Borges Sardinha

Orientadora: Ana Regina Cavalcanti da Rocha

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos - USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Área: Ciências da Computação e Matemática Computacional.

São Carlos, outubro de 1993

Ao meu filho Leonardo

AGRADECIMENTOS

À Prof^a Ana Regina C. da Rocha, um agradecimento especial por sua disponibilidade, que tornou possível a realização deste trabalho no Rio de Janeiro, e por sua orientação e apoio.

Aos professores da USP pelos ensinamentos, em especial ao Prof. Paulo César Mazieiro por sua colaboração.

À Laura e Beth, da secretaria da pós-graduação do ICMSC, por toda ajuda prestada durante este período em que estou distante do Campus de São Carlos.

Ao Núcleo de Computação Eletrônica da UFRJ pela disponibilização dos fontes do Pacote Estatístico PECS, em especial à Sonia Helt e ao Bourginom.

Aos colegas da USP e da COPPE pelo apoio, em especial à Tereza Aguiar, Jobson, Márcio, Rosângela e Vera Werneck.

Ao meus pais e ao Pedro pelo incentivo e apoio.

RESUMO

Este trabalho faz parte do Projeto TABA. O TABA visa a construção de uma estação de trabalho configurável capaz de criar um ambiente adequado às necessidades de cada projeto de desenvolvimento de software.

No contexto do Projeto TABA, um Ambiente de Desenvolvimento de Software (ADS) deve conter um conjunto integrado de métodos e ferramentas, que suportem as atividades de construção do produto, as atividades de planejamento e controle da qualidade e as atividades de gerência do processo de desenvolvimento.

Para dar suporte gerencial automatizado, imprescindível principalmente quando é grande a complexidade do sistema a ser desenvolvido, não existem, porém, ambientes capazes de suportar as diferenças existentes entre as características próprias de cada domínio de aplicação.

Esta tese apresenta uma solução para apoiar o gerente de projetos na configuração de uma estratégia gerencial adequada ao desenvolvimento de cada projeto, e que permite o acompanhamento e controle efetivos do projeto. Foi especificado um sistema com este objetivo, concebido como um conjunto de ferramentas integradas a uma base de conhecimentos, e desenvolvida uma das ferramentas que o implementa, um sistema de apoio à decisão baseado em conhecimento para o planejamento de recursos.

ABSTRACT

This work is part of the TABA Project. The main purpose of this project is the construction of a configurable workstation able to create an environment tailored to meet specific demands of each software development project.

In the TABA Project context, a Software Development Environment (SDE) should contain a set of integrated methods and tools, that support the construction activities, the quality planning and control activities and the development process management activities.

In order to provide an automatized managerial support, indispensable mainly whenever the complexity of the system to be development is high, we need environments able to support the differences between the particular characteristics of each application domain.

This work presents a solution to help the project manager in defining a managerial strategy tailored to a particular development project, and that allows the effective project monitor and control. A system was specified to meet this objective, conceived as a set of tools integrated with a knowledge base, and one of this tools was developed, a knowledge based support system for resources planning.

ÍNDICE

I) Capítulo I - Introdução	1
I.1) Objetivos do Trabalho	1
I.2) Organização do Conteúdo do Trabalho	8
II) Capítulo II - Gerência de Projetos	10
II.1) Visão Geral da Gerência de Projetos	10
II.2) Principais Atividades da Gerência de Projetos	14
II.2.1) Planejamento do projeto	14
II.2.1.1) Plano do Projeto	17
II.2.1.2) Aspectos Relacionados ao Planejamento do Processo de Desenvolvimento	19
II.2.1.3) Aspectos Relacionados ao Planejamento dos Recursos	21
II.2.1.4) Aspectos Relacionados à Organização da Equipe do Projeto	23
II.2.1.5) Aspectos Relacionados ao Planejamento do Acompanhamento do Projeto	31
II.2.1.6) Aspectos Relacionados ao Planejamento da Gerência de Riscos	34
II.2.1.7) Aspectos Relacionados ao Planejamento da Documentação	39
II.2.1.8) Aspectos Relacionados ao Planejamento da Gerência da Configuração	42
II.2.1.9) Aspectos Relacionados ao Planejamento do Controle da Qualidade	45
II.2.2) Acompanhamento do Projeto	47
II.3) Métodos de Gerência de Projetos	52
II.3.1) Modelo Integrativo e Dinâmico de Gerência de Projetos de Software	53
II.3.2) Modelos de estimativa de recursos	56
II.3.2.1) Modelo COCOMO	59

II.3.2.2) Pontos por função	70
II.3.2.3) Modelo de Putnam	78
II.3.3) Considerações sobre os Modelos de Estimativa de Custos	84
II.4) Ferramentas para Gerência de Projetos	97
II.5) Considerações Finais	106
III) CAPÍTULO III - Um Sistema de Gerência de Projetos de Desenvolvimento de Software para a Estação TABA	110
III.1) O Sistema de Gerência de Projetos no TABA	110
III.2) Requisitos do sistema	119
III.3) Especificação do sistema proposto	120
III.4) A Ferramenta de Planejamento de Recursos	128
IV) Capítulo IV - Conclusão	154
Referências Bibliográficas	158
Anexo I - Modelo proposto de Plano do Projeto (PPDS)	166
Anexo II - Descrição dos Processos e Dicionário de Dados .	176
Anexo III - Manual do Usuário	189

ÍNDICE DE FIGURAS

Fig. II.1: Sistema de Gerência de Software	53
Fig. II.2: Estrutura do Modelo	55
Fig. II.3: Estimativa de Tamanho e Custos de Software	94
Fig. II.4: Um Modelo de Estimativa de Tamanho e Custos de SW	96
Fig. II.5: Gerência do Projeto, dos Custos e dos Recursos	100
Fig. III.1: Ambientes do TABA	111
Fig. III.2: Diagrama de fluxo de dados do meta-ambiente ..	113
Fig. III.3: Ambientes TABA onde atua o sistema proposto ..	117
Fig. III.4: WBS do Sistema de Gerência de Projetos de Desenvolvimento de Software	121
Fig. III.5: DFD de Planejar Recursos (nível 0)	140
Fig. III.6: DFD de Estimar Custos (nível 1)	141
Fig. III.7: DFD de Planejar Organização (nível 1)	142
Fig. III.8: Estrutura modular do Sistema de Gerência	145
Fig. III.9: Estrutura modular de "PROCESSAR PLANEJAMENTO DOS RECURSOS" (Nível 1)	146
Fig. III.10: Estrutura modular de "PLANEJAR ORGANIZACAO" (Nível 2)	147
Fig. III.11: Estrutura modular de "PROCESSAR MODELO ESCOLHIDO" (nível 2)	148
Fig. A.III.1: Janela principal do Sistema de Gerência de Projetos de Desenvolvimento de SW	202
Fig. A.III.2: Janela principal da Ferramenta de Planejamento de Recursos	203
Fig. A.III.3: Janela para definição das características do do projeto	204
Fig. A.III.4: Janela para definição do modelo de estimativa a ser utilizado no projeto	205
Fig. A.III.5: Janela de apoio a escolha de modelo de estimativa de custos através da B.C.	206

Fig. A.III.6:	Janela para estimativa geral do tamanho e atributos do produto (Modelo COCOMO)	207
Fig. A.III.7:	Janela para estimativa geral do tamanho e atributos do produto (Modelo MACOS)	208
Fig. A.III.8:	Janela para estimativa geral do tamanho e atributos do produto (Modelo Pontos/Função)	209
Fig. A.III.9:	Janela para estimativa do tamanho e atributos dos componentes do produto (Modelo COCOMO) .	210
Fig. A.III.10:	Janela para estimativa do tamanho e atributos dos componentes do produto (Modelo MACOS) .	211
Fig. A.III.11:	Janela para estimativa do tamanho e atributos dos componentes do produto (Modelo Pontos/Função)	212
Fig. A.III.12:	Janela que exibe os resultados dos calculos das estimativas (Modelo COCOMO)	213
Fig. A.III.13:	Janela que exibe os resultados dos cálculos das estimativas (Modelo MACOS)	214
Fig. A.III.14:	Janela que exibe os resultados dos cálculos das estimativas (Modelo Pontos/Função)	215
Fig. A.III.15:	Janela para distribuição do esforço e tempo pelas atividades do projeto	216
Fig. A.III.16:	Janela para seleção do projetos semelhantes para acesso aos seus dados históricos	217
Fig. A.III.17:	Janela para exibir as características de semelhante selecionado	218
Fig. A.III.18:	Janela para exibir os dados de custo do projeto semelhante selecionado	219
Fig. A.III.19:	Janela para definir a estrutura organizacional adequada ao projeto	220
Fig. A.III.20:	Janela para alocação do pessoal às posições definidas na estrutura organizacional	221
Fig. A.III.21:	Janela para alocação do pessoal disponível aos níveis definidos	222
Fig. A.III.22:	Janela para exibição do "Plano de Recursos" elaborado através da ferramenta	223

ÍNDICE DE TABELAS

Tab. II.1: Fatores usados em modelos de estimativa de custo	58
Tab. II.2: Equações de esforço e tempo para o COCOMO Básico	60
Tab.II.3: Equações de esforço e tempo do COCOMO Intermediário	62
Tab.II.4: Coeficientes nominais correspondentes ao COCOMO original	68

CAPÍTULO I

INTRODUÇÃO

I.1) OBJETIVOS DO TRABALHO

Gerenciar projetos de desenvolvimento de software é sempre uma tarefa complexa e intrinsecamente difícil. Um produto de software é por si só um elemento complexo e sua produção pode envolver um esforço equivalente a centenas de homens-ano bem como um orçamento muito grande.

É comum em projetos de desenvolvimento de software ocorrerem numerosas dificuldades. Quando se diz que um projeto foi bem sucedido não é porque não ocorreram problemas mas porque estes problemas foram superados. A área de Engenharia de Software evoluiu de maneira significativa nos últimos anos, mas a realidade no desenvolvimento de software, hoje, ainda é a de se enfrentar grandes dificuldades técnicas, bem como orçamentos e cronogramas freqüentemente não cumpridos, provocando ceticismo entre os usuários dos produtos (Rocha [2]).

A maior causa da dificuldade de se ter um controle efetivo sobre projetos de desenvolvimento de software, segundo Keane [1], não é a falta de recursos técnicos para construção do software, e sim as falhas na aplicação de conceitos fundamentais de gerência, o que se dá por duas razões principais:

- a indústria de processamento de dados é altamente técnica e muito sujeita a mudanças rápidas, resultando na ênfase em treinamento técnico em detrimento de treinamento gerencial;

- a maioria dos gerentes de processamento de dados vieram da área técnica e tiveram poucas oportunidades de desenvolver

habilidades gerenciais.

Podemos, então, concordar com a afirmação de Rook [16]: "O que caracteriza um projeto bem sucedido é, justamente, o modo pelo qual ele é gerenciado".

Comparando gerência de desenvolvimento de software e gerência de desenvolvimento de hardware, que tem alcançado uma taxa muito maior de sucesso, observamos, em inúmeros aspectos, diferenças cruciais devidas principalmente às diferenças que o hardware apresenta em relação ao software:(Rook [16])

- o software não tem aparência física e inclui dados e lógica;
- existem poucas métricas de qualidade de software;
- o software é muito mais complexo que o hardware apesar de ser enganosamente fácil de se introduzir mudanças;
- os efeitos das mudanças no software se propagam de forma crescente; e
- o desenvolvimento de software faz pouco uso de componentes pré-existentes.

O mesmo autor pondera, entretanto, que existem similaridades que fazem com que o software deva ser gerenciado e controlado usando técnicas bastante parecidas às usadas pela engenharia de desenvolvimento de hardware. Estas diferenças fazem apenas com que a abordagem de engenharia deva ser muito mais rígida para o desenvolvimento de software.

O aumento da disciplina nas atividades técnicas de desenvolvimento tem como objetivo lidar com esses problemas e melhorar a confiabilidade do desenvolvimento de software. A ênfase da Engenharia de Software moderna é focalizar a atenção no

processo global de desenvolvimento. Este é o propósito de se dividir o projeto em fases, cada uma com metas bem definidas que podem ser verificadas se foram cumpridas para permitir o início da fase seguinte, e de se quebrar o trabalho a ser executado numa série de atividades gerenciáveis, o que permite criar a base para uma estrutura organizacional e planejar de forma global como o software será desenvolvido.

Partindo dessas questões, este trabalho tem como objetivo desenvolver um dos componentes da Estação TABA ((Rocha [9]) (Rocha [10])), a ferramenta para gerenciamento de projetos.

O projeto TABA, em desenvolvimento na COPPE/UFRJ (Programa de Engenharia de Sistemas e Computação), tem como objetivo a construção de uma Estação de Trabalho para o desenvolvimento de software, configurável para diferentes domínios de aplicação.

Analogamente a uma taba indígena, que suporta toda a vida de uma comunidade, a Estação TABA agrega várias ferramentas que suportam todo o ciclo de vida do software. A estação TABA pode, ainda, abrigar, ao mesmo tempo, diferentes ambientes construídos para projetos diversos.

A qualidade de produtos de software e a produtividade do seu processo de desenvolvimento dependem da qualidade e adequação do ambiente onde é desenvolvido.

Assim sendo, as características do ambiente de desenvolvimento devem ser determinadas considerando-se as características de cada aplicação. A definição deste ambiente, por sua vez, vai depender do estado da arte em métodos e ferramentas e do conhecimento de especialistas no uso dos mesmos.

A definição de um ambiente de desenvolvimento de software

adequado para cada área de aplicação ou projeto consome, entretanto, um esforço muito grande e exige um alto grau de conhecimento tanto no que se refere à engenharia de software quanto à área de aplicação do produto.

Assim, a Estação TABA deve atender aos seguintes objetivos:

- a) auxiliar o engenheiro de software no planejamento e criação do ambiente mais adequado ao desenvolvimento de um produto específico;
- b) auxiliar o engenheiro de software na implementação das ferramentas necessárias ao ambiente definido em (a);
- c) permitir aos desenvolvedores do produto (software) o uso da Estação através do ambiente especificado em (a) e gerado em (b);
- e
- d) permitir a execução do software, dado que, eventualmente, o produto poderá ser executado na própria Estação para ele configurada (o que é sempre verdade pelo menos na fase de teste).

Essas funções determinam quatro ambientes na estação TABA:

- a) Ambiente configurador de ambientes (meta-ambiente)

O objetivo do meta-ambiente é fornecer um acesso conveniente aos dados e conhecimentos auxiliando o engenheiro de software na configuração (especificação) do ambiente mais adequado ao desenvolvimento de um produto específico. A base de conhecimentos do meta-ambiente é alimentada tanto pelo ambiente gerador de ferramentas, quanto por informações sobre ferramentas e ambientes adquiridos externamente. Ao longo do desenvolvimento de um produto na Estação TABA, a base de conhecimentos do meta-ambiente sofrerá realimentação que irá influenciar na configuração de novos ambientes.

b) Ambiente gerador de ferramentas

Este ambiente vai auxiliar o engenheiro de software a implementar as ferramentas sugeridas pelo configurador, que passarão a fazer parte do repertório do mesmo a partir daí. Este ambiente é uma das instâncias do ambiente de desenvolvimento com especificidades por sua interação com o meta-ambiente.

c) Ambiente de desenvolvimento de software

A partir do ambiente específico definido, permite aos desenvolvedores do produto (software) usar a Estação.

d) Ambiente de teste e/ou execução da aplicação

Este ambiente permite executar o software que foi desenvolvido, na própria estação.

O meta-ambiente TABA tem duas funções principais: especificar o ADS mais adequado a um determinado projeto e torná-lo operacional.

A especificação do ADS é feita através de XAMÃ, o planejador de ambientes da Estação TABA (Aguiar [23]). Para sua implementação foram realizados vários estudos:

- aquisição do conhecimento de especialistas em Engenharia de Software (Glitz [29]);
- representação do conhecimento para métodos de Desenvolvimento de Software e mecanismos de inferência (Duarte [26]);
- ciclos de vida de software (Chitman [28]);
- taxonomia de ambientes de desenvolvimento (Moura [63]);
- taxonomia de áreas de aplicação (Werneck [27]);
- avaliação e seleção de métodos de desenvolvimento de software atualmente disponíveis (Crispim [25]); e,

- processo de desenvolvimento de sistemas baseados em conhecimento.

XAMÃ é um sistema baseado em conhecimento, que apóia o meta-usuário no planejamento do ambiente de desenvolvimento de software mais adequado a um determinado projeto.

Com o fim de tornar operacional o ADS especificado por XAMÃ, estão sendo realizados estudos e experimentos em integração de ferramentas.

No contexto do Projeto TABA, um ADS deve conter um conjunto integrado de métodos e ferramentas que suportem as atividades de construção do produto, as atividades de planejamento e avaliação da qualidade do produto (Rocha [62]) e as atividades de gerência do processo de desenvolvimento.

Dessa forma, para a completa configuração de um ADS a Estação TABA contém três módulos básicos: o módulo de construção, o módulo de avaliação da qualidade e o módulo de gerência.

O Módulo de Construção contém o conjunto operacionalizado das ferramentas de construção especificadas por XAMÃ.

O Módulo de Avaliação da Qualidade é composto de um conjunto de ferramentas integradas a uma base de conhecimentos. Estas ferramentas darão apoio ao desenvolvedor de um projeto no estabelecimento dos requisitos de qualidade de um produto e na avaliação dos produtos gerados em cada fase do processo de desenvolvimento. As ferramentas de avaliação são configuráveis para atender às características dos diferentes projetos, no que se refere a métodos de desenvolvimento e linguagens de programação.

O Módulo de Gerência, cuja definição é objeto deste

trabalho, é um conjunto de ferramentas integrado a uma base de conhecimento através do qual o gerente elaborará o Plano do Projeto e realizará o acompanhamento do processo de desenvolvimento.

A necessidade de se ter um módulo para o gerenciamento de projetos, no contexto de um ADS, fundamenta-se na necessidade de se fornecer um ambiente completo e integrado com suporte para todas as atividades envolvidas no processo de desenvolvimento de um software. Esta é uma tendência para os ADS's do futuro, especialmente quando os sistemas atingem uma complexidade que torna necessário o envolvimento de vários profissionais empenhados na tarefa de desenvolvê-los (Moura [63] Ghezzi [17]).

O Módulo de Gerência foi dividido em dois submódulos: Planejamento e Acompanhamento. O submódulo de Planejamento contém ferramentas para a elaboração do Plano do Projeto, integradas a uma base de conhecimento. Essas ferramentas geram, de acordo com o modelo de Plano do Projeto definido neste trabalho, o seguinte conjunto de planos:

- Plano do Processo de Desenvolvimento;
- Plano dos Recursos;
- Plano da Gerência do Projeto;
- Plano do Controle da Qualidade;
- Plano da Gerência da Configuração, e;
- Plano da Documentação.

O submódulo de Acompanhamento é composto por ferramentas para controle do progresso, acompanhamento e resolução dos riscos, revisão dos planos e encerramento e avaliação do projeto.

Dentro do escopo deste trabalho foi escolhida para

implementação a ferramenta de planejamento dos recursos, que conforme nosso modelo, envolve a estimativa dos recursos de pessoal, o planejamento da organização e alocação de pessoal, e a preparação dos cronogramas.

A implementação foi feita numa estação SUN, utilizando linguagem XVIEW/C e prolog, estando integrada a XAMÃ, e de forma a permitir a evolução do conhecimento implementado e o desenvolvimento futuro de outras ferramentas.

I.2) ORGANIZAÇÃO DO CONTEÚDO DO TRABALHO

Os demais capítulos deste trabalho estão organizados da seguinte forma:

O Capítulo II dá uma visão geral da área de gerência de projetos. Após considerações gerais sobre projetos de desenvolvimento de software abordando conceitos, dificuldades, a importância da gerência de projetos, e as responsabilidades e características de um gerente, são detalhadas as atividades da gerência: planejamento e acompanhamento. Por fim são discutidos os métodos e ferramentas disponíveis para apoio à execução dessas atividades.

O Capítulo III apresenta o módulo de Gerência de Projetos de Desenvolvimento de Software, localizando-o no contexto do Projeto TABA, e a ferramenta implementada para planejamento de Recursos.

No Capítulo IV são apresentadas as conclusões finais sobre a ferramenta desenvolvida, e sugeridas algumas idéias no sentido de aperfeiçoá-la em novas versões juntamente com o resto do ambiente de gerência proposto.

No Anexo I é apresentado o modelo proposto de Plano de Projeto de Desenvolvimento de Software, que foi utilizado como

base para definir as funções do módulo de gerência.

O Anexo II contém um Dicionário de Dados complementando a especificação de requisitos da ferramenta desenvolvida.

No Anexo III encontra-se o Manual do Usuário para utilização da ferramenta.

CAPÍTULO II

GERÊNCIA DE PROJETOS

A gerência de projetos é a execução repetida das seguintes atividades principais: planejamento das tarefas, organização dos recursos para executá-las, integração dos recursos e início da execução das tarefas, medição do realizado contra o planejado e revisão do plano, da organização e dos objetivos, quando ocorrerem desvios (P.Jones [7]) (Ghezzi [17]).

Neste capítulo é apresentada uma visão geral da gerência de projetos, abordando conceitos, dificuldades e destacando-se sua importância. São detalhadas as atividades envolvidas na gerência de projetos e, por último, são discutidos métodos e ferramentas para a sua realização.

II.1) VISÃO GERAL DA GERÊNCIA DE PROJETOS

Para tratar sobre gerência de projetos é importante que se defina primeiramente o que é um projeto. Assim, iniciaremos esta seção abordando alguns conceitos de projetos de desenvolvimento de software.

Podemos definir projeto, segundo Keane [1], como uma atividade específica com começo e fim, que produz resultados predeterminados. Qualquer projeto para ser bem sucedido deve ter um objetivo claro. Atingir este objetivo, porém, vai depender da disponibilidade de recursos adequados para completar a tarefa dentro do prazo e de habilidade para planejar, medir e controlar.

No caso particular de projetos de desenvolvimento de

software, sabemos que cada projeto é geralmente diferente e produz um produto único, requerendo uma coordenação de recursos única. Além disto, os participantes de projetos de software são constantemente designados a ambientes novos nos quais são encontradas necessidades e expectativas conflitantes por parte dos usuários, bem como uma constante demanda por mudanças, para atender a novas necessidades.

Entretanto, projetos desta natureza possuem características comuns, cujo entendimento pode ajudar bastante a racionalizar o processo de gerência. Entre estas características estão (Keane [1]):

- os produtos não são padronizados e, em geral, são caros;
- mudanças são freqüentes, mesmo durante o processo de produção;
- a coordenação das equipes de desenvolvimento é, freqüentemente, mais importante que máquinas e equipamentos;
- o pessoal é cedido apenas pela duração do projeto;
- a produtividade é difícil de ser controlada e medida, dependendo bastante da motivação do pessoal;
- o projeto envolve usuários que devem tomar decisões mas que, freqüentemente, possuem pouca ou nenhuma experiência em participar de projetos de desenvolvimento de software, além de nem sempre entenderem totalmente seus papéis no projeto;
- o gerente do projeto não tem autoridade direta sobre muitos dos recursos que ele necessita para executar seu trabalho.

Estas características levam à recomendação que se divida o trabalho em etapas para tornar o projeto mais facilmente gerenciável. O número de etapas não importa, o que deve ser

levado em conta é que os membros da equipe do projeto concordem entre si e com os usuários em como o processo de desenvolvimento será segmentado e o que deve ser feito para completar cada etapa. Uma vez segmentado o trabalho, cada etapa deve ser considerada como um projeto por si só, já que cada uma é uma atividade bem definida com começo e fim, e produz um resultado predeterminado (Pressman [3]) (Presman [64]) (Keane [1]).

Além disto, as características particulares de projetos de desenvolvimento de software determinam uma série de responsabilidades ao gerente de projetos. A primeira delas é a escolha de um ambiente de desenvolvimento adequado para ser usado no projeto. Este ambiente deve ser composto, segundo Aguiar [23], pelo ciclo de vida adotado para o projeto e pelos métodos, ferramentas e hardware necessários para sua execução.

Um ambiente de desenvolvimento por si só não garante, porém, a qualidade do produto nem o sucesso de um projeto. É necessária uma abordagem gerencial sistemática e cuidadosa no que se refere ao planejamento e controle do processo de desenvolvimento.

Com este objetivo vários autores sugerem um conjunto de atividades classificadas segundo diferentes esquemas, como Ghezzi [17], Boehm [55], Rook [16] e Keane [1]. A proposta mais recente é a de Ghezzi [17], que define como principal atividade do gerente: "Habilitar um grupo de pessoas a trabalhar em direção a um objetivo comum". Ele classifica como padrão cinco atividades interrelacionadas, cujos objetivos são atingir efetivamente um trabalho em equipe. São elas:

- **Planejar:** Consiste em determinar os objetivos e os recursos para atingí-los, como e quando adquirir os recursos e

como os objetivos serão atingidos.

- **Organizar:** Envolve escolher a forma de organização e definir linhas claras de autoridade e responsabilidade para os grupos de atividades.

- **Alocar:** Trata da alocação do pessoal para as posições identificadas na estrutura organizacional. Envolve definir os requisitos de pessoal, recrutar, recompensar, desenvolver e promover o pessoal.

- **Dirigir:** envolve liderar os subordinados de forma a fazê-los entender e se identificar com a estrutura organizacional e com os objetivos do projeto.

- **Controlar:** Consiste em medir e corrigir atividades para assegurar que os objetivos sejam alcançados. Requer medir o desempenho contra os planos e tomar ações corretivas quando ocorrerem desvios.

Estas atividades gerenciais não implicam, porém, em centralização. O gerente pode, e deve, delegar autoridade e poder de decisão, mas deve ficar com a responsabilidade final por aspectos como: relacionamento com o cliente, escolha das ferramentas para gerenciar o projeto, obtenção dos recursos, especificação, exatidão do projeto e da implementação, qualidade, uso de recursos e gerência da equipe de desenvolvimento, obediência ao cronograma, ao orçamento e aos padrões e procedimentos estabelecidos, tratamento de problemas que possam surgir, entrega final e aceitação do produto. (Rook [16])

Todas essas responsabilidades tornam indispensáveis ao gerente possuir habilidade e experiência em planejamento, organização, controle e liderança de projetos. Um gerente de

projetos deve preencher três requisitos para ser um líder efetivo, segundo Pulk [34]: ter habilidade para gerenciar projetos, ter habilidade para desenvolver software e ter conhecimento do produto a ser desenvolvido.

Segundo Boehm [21], uma gerência inadequada faz os custos do software aumentarem mais que qualquer outro fator. Muitas vezes, porém, se tem que contar com um gerente inexperiente ou sem as qualidades desejadas, pelos mais diversos motivos. O êxito do projeto nem por isto estaria comprometido, desde que se dê a devida importância ao planejamento e controle. (Rocha [2])

II.2) PRINCIPAIS ATIVIDADES DA GERÊNCIA DE PROJETOS

A partir das abordagens citadas na seção anterior e de outras como Rocha [2], Pressman [3] e Pressman [64], consideramos neste trabalho a gerência do processo de desenvolvimento como estando dividida em duas atividades fundamentais: planejamento e acompanhamento. A seguir discutiremos aspectos relacionados à realização destas atividades, dando ênfase ao planejamento e seu produto, o Plano do Projeto, uma vez que o acompanhamento consiste basicamente em executar o que foi planejado, controlando os desvios.

II.2.1) PLANEJAMENTO DO PROJETO

O planejamento é imprescindível ao sucesso de um projeto de desenvolvimento de software porque sem ele estamos sujeitos a atrasos, a ultrapassar o orçamento e a produzir um produto de baixa qualidade levando à insatisfação dos usuários.

Atrasos e orçamentos ultrapassados são uma constante em

projetos de desenvolvimento de software e têm três categorias de motivos principais, segundo Keane [1]:

- estimar antes de se ter informações suficientes;
- ceder às imposições de gerentes quanto ao prazo e/ou orçamento, sem avaliar a razoabilidade destas imposições;
- não incluir na estimativa todos os fatores que influenciam nos custos ou não avaliá-los corretamente.

Entre os muitos fatores que dificultam a tarefa de estimar recursos estão (DeMarco [5]) (Keane [1]) (Cuelenaere, Genuchten e Heemstra [11]):

- o número de fatores que influenciam nos custos, esforço e duração de um projeto de software, a falta de definição clara sobre esses fatores, seus interrelacionamentos e a dificuldade de avaliar seu impacto;
- falta de dados históricos para calibrar softwares de estimativa de custo e escassez de especialistas;
- práticas tendenciosas;
- falha na avaliação do ambiente do projeto, e;
- volatilidade dos requisitos.

Como conseqüências de um planejamento sem base e um início prematuro da programação temos: replanejamento freqüente, emprego inadequado dos recursos, necessidade de esforço desproporcional para teste e correção de erros, inabilidade em determinar quando o projeto estará completo, insatisfação do usuário e aumento dos custos de manutenção (Keane [1]) (Pulk [34]).

O planejamento sem base, no início, pode parecer um ganho de tempo mas durante os testes o engano é percebido: encontra-se um elevado número de erros e os usuários descobrem que os resultados

não são os esperados. Este problema é conhecido como a síndrome dos 90%: os últimos 10% do projeto parecem conter 90% dos problemas (Brooks [46]).

O planejamento criterioso embora não garanta que um projeto tenha êxito, propiciará que os problemas que surgirem possam ser superados com muito mais facilidade (Rocha [2]).

Para planejar, porém, o gerente de projetos se confronta com um dilema logo no início do projeto. O planejamento envolve estimar a complexidade do produto, a produtividade individual do pessoal de desenvolvimento, a disponibilidade de pessoal e outros recursos, etc. Para isto, medidas quantitativas são requeridas, mas não existem informações sólidas neste estágio. Uma análise detalhada dos requisitos do software poderia prover as informações necessárias, mas a análise geralmente leva semanas ou meses para ser completada (Pressman [3]) (Pressman [64]) (Ghezzi [17]).

Para contornar este problema o gerente deve contar com a experiência de especialistas, ter acesso a informações históricas adequadas e revisar regularmente, com o progresso do projeto, as estimativas feitas no início. Medidas compreensíveis de projetos de software passados podem dar maior segurança a estimativa, cronogramas podem ser estabelecidos de forma a evitar dificuldades, e o risco global pode ser reduzido.

O produto da atividade de planejamento é o Plano do Projeto, que se constitui no conjunto organizado dos elementos definidos no planejamento. É de pouca utilidade possuir um conjunto de estimativas desordenadas e difíceis de manipular (Rocha [2]).

A seguir discutiremos de forma breve sobre as características que um plano de projeto deve possuir e detalharemos os aspectos indispensáveis que devem ser abordados em um Plano de Projeto de desenvolvimento de software.

II.2.1.1) PLANO DO PROJETO (Rocha [2]) (Pressman [3]) (Pressman [64])

Em forma de um documento relativamente breve, elaborado formalmente, o Plano do Projeto é a ferramenta básica que deverá ser utilizada para guiar o projeto em direção a seus objetivos.

É o Plano do Projeto que comunica para a gerência de desenvolvimento de software, para a equipe técnica e para o cliente, o escopo do projeto e os recursos necessários, dando uma visão geral da abordagem de desenvolvimento de software adotada. Desta forma, ele deverá concentrar-se em questões como O QUE, QUANTO CUSTA, QUANTO DURA.

Como o planejamento é uma atividade contínua ao longo do desenvolvimento, o Plano do Projeto deve ser manutenível, permitindo ser atualizado e detalhado ao longo do processo de desenvolvimento.

Ao terminar de preparar o plano deve-se estudar a viabilidade de sua operação, sob os pontos de vista financeiro, técnico, administrativo e de recursos de suporte. Sendo viável, o plano deve ter o aceite do usuário. Antes da aprovação autorizando prosseguir com o projeto, nenhum trabalho deve ser iniciado, sob pena de ter que ser refeito.

Quanto à organização do plano, muitos roteiros são propostos na literatura que podem ser adaptados para uso. Entre eles

destacamos o de Rocha (Rocha [2]), os de Boehm (Boehm [4], [55]), o modelo do IEEE (IEEE [54]), o de Evans (Evans [56]), o de Metzger (Metzger [57]) e o de Pressman (Pressman [3] e Pressman [64]).

Estes roteiros diferem não apenas na forma de organização, como também nos aspectos do planejamento incluídos e no grau com que são voltados para um determinado tipo de ciclo de vida. Baseados nesses modelos consideramos, neste trabalho, que o Plano deve conter os seguintes aspectos:

- a) **Planejamento do processo de desenvolvimento**, que consiste na definição detalhada do trabalho a ser feito através da identificação dos produtos a serem gerados, das tarefas a serem efetuadas e de marcos de controle.
- b) **Planejamento dos recursos**, que envolve estimar o custo dos recursos de pessoal, de hardware, de software (por exemplo: ferramentas) e a duração das atividades.
- c) **Planejamento da organização da equipe do projeto**, que envolve determinar a forma da estrutura organizacional do projeto e promover a aquisição e integração dos recursos.
- d) **Planejamento do acompanhamento do projeto**, que consiste em planejar os mecanismos e procedimentos que serão utilizados para fazer a monitoração e controle da evolução do projeto à luz dos planos e marcos que foram estabelecidos.
- e) **Planejamento da gerência dos riscos**, que envolve identificar os riscos do projeto, avaliar seu impacto e preparar um plano de contingência para monitorá-los e controlá-los ao longo do projeto.
- f) **Planejamento da documentação**, que consiste em definir os procedimentos para a geração e publicação dos documentos a serem

produzidos bem como o estilo desses documentos.

g) **Planejamento da gerência da configuração**, que consiste em determinar o conjunto de atividades de rastreamento e controle necessários para manter a correspondência entre as diversas representações da forma não executável do software, garantindo que expressem o pretendido.

h) **Planejamento do controle da qualidade**, que envolve determinar o conjunto sistemático de ações necessárias para fornecer adequada garantia de que o item ou produto está de acordo com os requisitos estabelecidos.

Além desses, o planejamento de outros aspectos pode ser necessário também, dependendo de sua importância para o projeto, como por exemplo: planejamento da gerência de subcontratados, considerações de segurança, planejamento do treinamento, planejamento da instalação, planejamento da conversão de dados, planejamento da transição do sistema e planejamento da manutenção do produto.

A realização dessas atividades de planejamento envolve uma série de questões que devem ser consideradas a partir das características particulares de um determinado projeto. Existem várias considerações e propostas na literatura que detalharemos a seguir.

II.2.1.2) Aspectos relacionados ao planejamento do processo de desenvolvimento

A definição detalhada do trabalho a ser feito é um processo interativo cuja definição começa no início do ciclo de vida do projeto e se torna mais detalhada com o progresso do projeto.

Para planejar o projeto é necessário definir o escopo do sistema, especificando explicitamente os produtos a serem gerados, baseado nas funções e desempenho aplicados ao software proposto. O escopo deve ser limitado e tratar explicitamente dos dados quantitativos, das restrições e/ou das limitações. As informações necessárias para descrever o escopo podem ser retiradas da documentação de especificação do sistema, elaborada na fase de análise de requisitos (Pressman [3] Pressman [64]).

Para definir as atividades necessárias à geração dos produtos e os marcos de controle, deve ser escolhido e/ou adaptado um modelo de ciclo de vida adequado ao projeto. Vários fatores determinam o perfil do ciclo de vida, como: tipo de linguagem de programação utilizada, suporte gerencial que se pretende adotar, recursos que possibilitem a reutilização, tipo de aplicação e o fato do ciclo ser para desenvolvimento a partir do início da vida do software ou para manutenção. De maneira geral as atividades do ciclo de vida atendem: a construção do software e a sua documentação, a verificação da qualidade, testes, depuração e reparo de erros, e a objetivos gerenciais (Aguiar [23]).

A partir da definição do ciclo de vida o gerente do projeto poderá, então, selecionar um conjunto de métodos, ferramentas e o hardware que suportem o desenvolvimento do software nas diferentes etapas desse ciclo. O ciclo de vida, métodos, ferramentas e o hardware são os componentes a partir dos quais é formado o ambiente de desenvolvimento de software (ADS) (Aguiar [23]).

II.2.1.3) Aspectos relacionados ao planejamento dos recursos

A estimativa dos recursos pode ser considerada como o aspecto mais crítico do planejamento. Hoje, o custo do software, primariamente de pessoal, é o maior componente do custo total de um sistema computadorizado. Assim, o sucesso financeiro de um projeto pode depender da habilidade para estimar precisamente, antes do projeto começar, o custo dos recursos de pessoal necessários para o desenvolvimento do software (Pressman [3] Pressman [64]) (Ghezzi [17]).

Por esse motivo neste trabalho consideramos o custo dos recursos como sendo apenas custo de pessoal devido a seu peso em relação a recursos de software e hardware.

As medidas de produtividade obtidas na produção formam a base para estimar os recursos requeridos durante o planejamento e para, mais tarde, avaliar o desempenho de indivíduos, processos, ferramentas, benefícios da automação, etc.

As unidades básicas a partir das quais a produtividade é medida são usadas de duas maneiras durante a estimativa: como variável de estimação, que é usada para dimensionar cada elemento do software, e como métricas base coletadas de projetos passados e usadas junto com as variáveis de estimação para fazer projeções de custo e esforço (Ghezzi [17]).

A avaliação dos dados de produtividade de projetos passados deve focar as razões fundamentais para os resultados obtidos, ponderar se as medidas calculadas são relevantes para o novo projeto e preocupar-se com circunstâncias que invalidam certos dados para uso na estimativa (Pressman [3] Pressman [64]).

Para ter medidas significativas, que permitam assegurar se

houve ganho de produtividade, necessita-se medir a produtividade quantitativamente. Para isto são requeridas métricas. Uma métrica ideal de produtividade deve medir não linhas de código, mas a quantidade de funcionalidade produzida por unidade de tempo. A dificuldade de quantificar o conceito de funcionalidade, porém, levou ao desenvolvimento de várias métricas de produtividade baseadas no tamanho do produto mais tangível do desenvolvimento, o código produzido. As duas métricas mais comuns de tamanho do código são DSI ("delivered source instructions") e NCSS ("non-commented source statements"). A unidade genérica, porém, é KLOC ("thousands of lines of code") (Ghezzi [17]).

As métricas de produtividade existentes são controvertidas. Sua adoção e uso devem ser feitas cuidadosamente e seus resultados não devem ser usados como medidas objetivas da habilidade ou produtividade de uma pessoa. Além disto, independentemente da métrica usada para produtividade, deve-se avaliar, também, os fatores que afetam a produtividade (Ghezzi [17]).

Métricas de produtividade permitem quantificar os benefícios de vários fatores. Os efeitos destes fatores na produtividade são refletidos nos atributos de custo usados nos modelos de estimativa de custo. Para estimar o custo necessita-se de métodos capazes de estimar a complexidade do software antes dele ter sido desenvolvido, baseados apenas nas informações disponíveis, como os requisitos (Ghezzi [17]).

Hoje dispomos de vários métodos para estimar custos. Nenhum deles, porém, é melhor que os outros e o ideal é utilizar uma combinação destes, de forma que se obtenha as vantagens

individuais de cada um e se minimize as desvantagens, e depois comparar e interagir as estimativas obtidas de cada um. Os métodos de estimativa de custos mais conhecidos são descritos na Seção II.3.

II.2.1.4) Aspectos relacionados à organização da equipe do projeto

A função gerencial de definir a organização das pessoas diretamente envolvidas no projeto tem como principais objetivos a divisão das tarefas entre as pessoas e a atribuição de responsabilidades para atingir as metas do projeto.

É comum, na maioria das organizações de informática, a utilização de um único arranjo organizacional, padrão, para tratar com projetos distintos em termos de importância para a empresa, tecnologia utilizada, tamanho, etc.

A definição da estrutura organizacional a ser adotada no projeto transcende os seus aspectos meramente técnicos. O fator humano deve ser considerado no planejamento psicologicamente, individualmente e em grupo, bem como sua influência no desempenho da equipe. Junto com as características particulares de projetos de desenvolvimento de software, ele nos leva a escolha da estrutura organizacional apropriada e a maneira como essa organização pode ajudar na obtenção de um software de qualidade.

Entre as questões que afetam a escolha da organização destacamos (Ghezzi [17]) (Kugler [65]):

- a duração do projeto;
- a natureza das tarefas e a quantidade de comunicação entre os membros da equipe que elas requerem;
- o tamanho da equipe;

- a orientação do trabalho, se voltado para eficiência ou eficácia;
- capacidade de persuasão do gerente;
- conflitos de autoridade, e;
- liderança.

A organização de equipe pode ter um controle centralizado, onde um líder reconhecido é responsável por autorizar e resolver todas as questões técnicas, ou pode ser baseada na distribuição do controle das decisões e enfatizar o consenso do grupo.

A avaliação do melhor tipo de estrutura organizacional é difícil e a comparação de resultados obtidos com diferentes tipos de organização é complexa. Assim, é importante o gerente do projeto avaliar algumas considerações antes de escolher a organização da equipe, entre as quais destacamos (Ghezzi [17]) (Rocha [2]):

- Nenhuma organização de equipe é apropriada para todas as tarefas;

- Uma organização apropriada é aquela que tenta limitar a quantidade de comunicação à estritamente necessária para atingir as metas do projeto. Ela elimina mal entendidos que ocorrem muitas vezes quando as pessoas trabalham isoladas, e encoraja revisões, aumentando a qualidade do software produto;

- A organização apropriada deve suportar outras metas além de velocidade do desenvolvimento, como: ciclo de vida de baixo custo, rotatividade reduzida de pessoal, disseminação do conhecimento e promover os profissionais menos experientes;

- O controle descentralizado é melhor quando a comunicação entre os membros se faz necessária para que se tenha uma boa

solução.

- O controle centralizado é mais indicado quando a velocidade do desenvolvimento é a meta mais importante e o problema esta bem entendido;

- Como resultado de pesquisa realizada em empresas brasileiras por Alencar, Rocha e Silva (Rocha [2]) foi verificado que a produtividade aumentava de acordo com o tipo de organização na seguinte ordem: controle descentralizado, equipes convencionais, equipes hierárquicas e time do programador chefe.

Os tipos de organizações, classificados pelo tipo de controle decisório, são (Ghezzi [17]) (Rocha [2]):

a) Equipe convencional

Se caracteriza por ser composta pelo pessoal disponível no início do projeto, não existindo preocupação com a estrutura organizacional e o perfil das pessoas. O pessoal fica sob a responsabilidade de um gerente de desenvolvimento designado para o projeto e cada um é responsável por uma parte do projeto, desde o início até à implementação, o que cria um forte sentimento de posse e uma tendência a não se reconhecer os próprios erros, entre outros problemas.

b) Organização com controle centralizado

A organização com controle centralizado é baseada na estrutura hierárquica onde várias pessoas reportam-se a um supervisor, que controla diretamente suas tarefas e é responsável por seus desempenhos. As comunicações são descentralizadas nos subgrupos e centralizadas nos níveis mais altos. Funciona bem para tarefas simples o bastante que uma pessoa, o responsável por controlar o projeto, possa dominar o problema e a solução.

c) Organização com controle descentralizado

As decisões são consensuais e todo o trabalho é considerado do grupo. Os membros da equipe revisam mutuamente seus trabalhos e são responsáveis como um grupo pelo que todos os membros produzem. Aqui não existe hierarquia e todos os membros estão no mesmo nível. A organização democrática leva a uma moral elevada, à satisfação com o trabalho, que minimiza a rotatividade, e a uma maior responsabilidade pelo problema, o que leva a um grau maior de qualidade.

Este tipo de organização é mais adequada para projetos complexos, uma vez que um grupo pode criar uma solução melhor que a de apenas um indivíduo. Não é apropriado, porém, para equipes grandes devido à sobrecarga de comunicação que reduz a produtividade individual.

d) Organização de equipe com controle misto

Este tipo de organização pretende combinar os benefícios dos controles centralizado e descentralizado, enquanto minimiza ou evita suas desvantagens. Ela distingue os profissionais em "seniors" e "juniors", e cada "senior" gerencia um grupo de "juniors". Os "seniors" se reportam ao gerente do projeto e ambos detêm o controle. A comunicação é descentralizada entre cada conjunto de indivíduos pares e seus supervisores imediatos, limitando-se, dessa forma, a um pequeno grupo. Investindo a autoridade num grupo de programadores "senior", obtém, também, os benefícios da decisão em grupo.

Outra classificação é dada em Kugler [65], agrupando os arranjos organizacionais em três classes: (a) abordagens clássicas; (b) variações das abordagens clássicas; (c) outras

abordagens.

a) Abordagens clássicas

Estrutura funcional pura

- Características:

- Especialização/departamentização por funções;
- Ênfase/orientação - "tarefa"
- Unidade de comando fortemente calcada na hierarquia;
- Medida de avaliação do desempenho gerencial - eficiência;
- Fonte de poder - autoridade formal;
- Geralmente observada em pequenos centros de processamento de dados.

- Vantagens:

- Adequada para conduzir projetos relativamente pequenos ou pouco complexos;
- Facilita a avaliação de desempenho do pessoal alocado em termos de eficiência;
- Propicia maior disciplina administrativa e facilita a padronização de procedimentos.

- Desvantagens:

- Cada gerente funcional visa seu lado em detrimento de uma ação conjunta com vistas a objetivos comuns;
- Facilita o artifício de jogar o problema para as outras gerências quando ocorrem impasses, já que a responsabilidade pelos resultados do projeto é bastante difusa;
- Torna difícil a avaliação de desempenho do pessoal alocado em termos de eficácia;
- Torna difícil o planejamento, a coordenação, a integração e o acompanhamento de cada projeto;

- Torna difusa a responsabilidade pelos resultados finais de cada projeto;

- Gera problemas de comunicação.

Estrutura funcional/projetos

- Características:

- Equipes multidisciplinares, formadas em função de um projeto e não em função de "tarefas";

- Alocação dinâmica de recursos;

- Ênfase/orientação - "missão";

- Medida de avaliação do desempenho gerencial - eficácia;

- Fonte de poder - influência, liderança, "expertize";

- Participantes do projeto alocados diretamente sob o líder de projeto (pelo menos os técnicos de execução);

- Estrutura observada na maioria dos centros de informática no Brasil.

- Vantagens:

- Adequada para conduzir projetos grandes, que requerem bastante tempo e atenção por parte de seu responsável;

- Os recursos são parcialmente alocados ao gerente de projeto, facilitando direcionamento e controle;

- Os problemas são detectados mais facilmente, possibilitando ações corretivas mais rápidas;

- Existe centralização gerencial em torno do projeto, facilitando o acompanhamento.

- Desvantagens:

- Exige grande capacitação gerencial;

- Dificulta a avaliação global de pessoal em relação ao plano de cargos e salários;

- Dificulta o intercâmbio de experiência técnica e profissional com participantes de outros projetos;
- Gera conflitos de autoridade entre projetos concorrentes pelos mesmos recursos e com as gerências funcionais;
- Os gerentes funcionais relutam em ceder seu pessoal para as equipes de projeto.

Estrutura matricial/balanceada

- Características:

- Bastante semelhante a "por projetos";
- Medida de avaliação do desempenho gerencial - eficácia;
- Fonte de poder - liderança, "expertise", mas fortemente dependente do engajamento de terceiros com relação ao projeto;
- "Participantes" do projeto divididos em técnicas de execução, consultores/colaboradores;
- Adequada para projetos grandes, envolvendo várias tecnologias de informação simultaneamente.

b) Variações das abordagens clássicas

Estrutura matricial/funcional

- Características:

- Abaixo do gerente funcional há gerentes de projetos integrados, que utilizam colaboração de técnicos de outras áreas;
- Conserva as mesmas características da matriz/balanceada;
- Adequada para projetos grandes, envolvendo várias tecnologias de informação simultaneamente.

Estrutura matricial/projetos

- Características:

- O gerente do projeto tem nível hierárquico superior ao gerente funcional;
- Aplicável para projetos grandes, envolvendo várias

tecnologias e de grande prioridade para a empresa;

- Conserva as mesmas características da matriz/balanceada.

c) Outras abordagens

Estrutura por cliente

- Características:

- Facilita a comunicação com usuários devido à especialização;
- Dificulta a avaliação/mensuração de resultados.

Estrutura por linha de produto

- Características:

- Facilita a comunicação com os responsáveis pelo produto.

Estrutura por tipo de sistema

- Características:

- Sistemas divididos em:
 - Sistemas corporativos (Recursos Humanos, Contabilidade, Orçamento, Vendas, Planejamento, etc);
 - Sistemas Específicos por unidade industrial.

Escolhida a forma da estrutura organizacional e selecionadas as habilidades requeridas nesta, o gerente do projeto deve avaliar o perfil do pessoal disponível, para verificar se atendem ao nível mínimo exigido pelo projeto. Na falta de pessoal mais qualificado a equipe pode ser composta por pessoas que possam produzir o resultado requerido, embora num tempo maior.

A correta alocação de pessoal ao projeto impulsiona a produtividade, evita o aumento nos custos provocados por se adicionar pessoas em fases adiantadas do projeto e possibilita fazer análises de custos versus tempo de desenvolvimento.

Tratando-se de elemento humano, especificar e obter o

recurso não são as únicas tarefas. É de suma importância que se obtenha o envolvimento dos usuários e do pessoal técnico apropriado para: estabelecer uma comunicação efetiva entre todos os envolvidos no projeto, obter a necessária cooperação e envolvimento dos usuários, e motivar a equipe a atingir os objetivos do projeto (Keane [1]).

II.2.1.5) Aspectos relacionados ao planejamento do acompanhamento do projeto

Quanto mais cedo os desvios do plano forem detectados maior a chance de controlá-los. Para isto, precisa-se planejar os mecanismos e procedimentos que serão utilizados. Basicamente dois tipos de mecanismos podem ser identificados: mecanismos para relato e mecanismos para monitoração e controle.

O planejamento dos mecanismos de relato envolvem a definição do tipo (oral ou escrito), formato, frequência e distribuição dos relatórios.

Os mecanismos de monitoração e controle consistem de ferramentas de programação e controle de atividades e de planilhas e gráficos para planejamento da distribuição das atividades, do orçamento e dos recursos e para registro da evolução do progresso e do consumo de recursos durante o projeto.

As ferramentas de programação e controle de atividades são baseadas em técnicas de controle. A maioria dessas técnicas consiste em decompor os objetivos do projeto em objetivos menores que são refinados até um tamanho que permita o bom entendimento de cada um. Para cada objetivo individual pode-se, então, planejar seus requisitos de recursos, atribuir responsabilidades, preparar cronogramas, etc. Uma maneira semi-formal de

decomposição muito conhecida é chamada de "work breakdown structure" (WBS).

Na técnica WBS (Ghezzi [17]) constrói-se uma árvore cuja raiz é rotulada com a atividade principal do projeto. Cada nó da árvore vai sendo decomposto em componentes menores até que os nós do nível mais baixo da hierarquia possam ter seu tamanho, dificuldade e requisitos de recursos facilmente estimados.

Essa técnica é simples e permite que as tarefas possam ser decompostas em unidades mais facilmente gerenciáveis que podem servir de base para atribuir trabalho. A estrutura também pode ser refinada e estendida facilmente, adicionando-se informações nos nós como: duração planejada da atividade, nome do responsável e datas de início e término da atividade. Além disto, serve como entrada para o processo de preparação dos cronogramas.

Decompondo-se o trabalho, tenta-se decidir que tarefas precisam ser feitas, enquanto que, preparando-se cronogramas se decide a ordem na qual estas tarefas serão feitas para assegurar o cumprimento do prazo. A partir dos cronogramas é possível planejar pontos de controle ao longo do projeto marcando revisões de avaliação e aceite dos produtos gerados.

Existem várias técnicas gráficas para documentar cronogramas, sendo as mais consagradas a rede de PERT e o gráfico de Gantt.

Redes de PERT (Ghezzi [17]) são redes compostas de retângulos ou círculos, que representam as atividades e sua duração, e de setas, que representam a relação de precedência entre elas. A atividade da extremidade final da seta só pode

começar quando a atividade da extremidade inicial estiver concluída.

Da mesma forma que para o WBS, os retângulos podem receber informações como data de início e término e recursos requeridos. As setas ajudam no cálculo da data de início mais cedo possível para as atividades da sua extremidade inicial. Além disto, alguns retângulos podem, ainda, ser designados como marcos.

Para que se possa saber qual o tempo mínimo para conclusão do projeto e que atividades são críticas para se conseguir concluir o projeto no menor tempo possível, determina-se o caminho crítico, que significa o caminho mais longo através da rede em termos de duração total das atividades. Um atraso qualquer de uma atividade do caminho crítico atrasará o projeto no tempo correspondente

Um projeto pode ser encurtado, encurtando-se seu caminho crítico, no caso de existir apenas um, ou encurtando atividades comuns a todos os caminhos críticos do projeto, no caso de existir mais de um.

Além do caminho crítico, é útil determinar o momento mais tarde que se pode iniciar cada atividade sem se atrasar o término geral do projeto, e qual a duração máxima que uma atividade pode ter, no caso de se iniciá-la o mais cedo possível, de forma a não atrasar o término geral do projeto. A duração máxima pode ser usada para atrasar o início de uma atividade ou para estender sua duração (ou ambos).

Gráficos de Gantt (Ghezzi [17]) é uma técnica de preparo de cronogramas para vários propósitos, sendo os mais comuns: cronograma de atividades, cronograma financeiro e cronograma de

planejamento de uso de recursos. No gráfico de Gantt as atividades são representadas como linhas horizontais numa planilha, cujas colunas representam os meses (ou outra unidade de tempo) e triângulos indicam os principais marcos associados com a atividade. O progresso do projeto é indicado preenchendo-se estes triângulos quando o marco é alcançado.

O gráfico de Gantt é uma técnica simples e fácil de usar, e provê informações sobre a situação do projeto e sobre a distribuição das atividades. Porém, não oferece meios de mostrar as relações de dependência entre as atividades, o que pode levar a ambigüidades.

Gráficos de Gantt podem ser derivados automaticamente de gráficos de PERT e muito auxiliam no planejamento da utilização dos recursos. Já os gráficos de PERT são melhores para monitorar o progresso das atividades.

II.2.1.6) Aspectos relacionados ao planejamento da gerência de riscos

De acordo com Wideman [52], a gerência de riscos de projetos é "a arte e ciência de identificar, analisar e responder a situações de risco ao longo da vida do projeto". O principal objetivo da gerência de riscos para projetos de software é evitar que os problemas que inevitavelmente aparecem nesses projetos, se tornem crises. Para isto é importante que o gerente do projeto tenha conhecimento de técnicas para identificar os riscos do projeto, avaliar seus impactos, monitorá-los e controlá-los.

Até o início da década de 80 a gerência de riscos para projetos de software vinha sendo conduzida de maneira informal.

Ela começou a ser formalizada com o surgimento de trabalhos como: Boehm [49], AFSC [48], DSMC [50], NSIA [51], Fairley [47] e Safavi [72].

Um programa sistemático de gerência de riscos pode ser dividido em duas partes principais: avaliação dos riscos e o controle dos riscos. A avaliação dos riscos consiste da identificação dos riscos, da análise de seu impacto potencial e da priorização dos riscos. O controle dos riscos consiste do desenvolvimento de planos de contingência, do estabelecimento de procedimentos de monitoração dos riscos, da implementação dos planos de contingência, da gerência da recuperação dos riscos quando necessário, e da tomada de ações em resposta a situações de crise não previstas (Fairley [47]).

Como os produtos de software são artefatos invisíveis sujeitos a restrições lógicas, a gerência de riscos para projetos de software foca-se no processo usado para desenvolver estes artefatos. Além disto, como esses processos são primariamente humanos, os fatores e as probabilidades de risco são determinados pela análise de dados históricos e por julgamento subjetivo de especialistas. Análise de sensibilidade (análises "what if") é uma importante ferramenta para determinar o impacto de possíveis desvios das tendências históricas e de erros de julgamento de especialistas (Fairley [47]).

Os principais riscos inerentes a um projeto são: não produzir o produto desejado; consumir mais recursos que o designado no seu orçamento; e gastar mais tempo que o determinado nos cronogramas (Ghezzi [17]).

Apesar de muitos fatores poderem causar estes riscos, alguns

problemas ocorrem com tanta freqüência, que se tornaram padrões identificáveis. De acordo com Brooks [46], cronogramas irreais é o fator predominante, enquanto que Boehm [49] coloca os fatores de pessoal em primeiro lugar. Fairley [47] apresenta uma tabela com as cinco maiores áreas de risco para um projeto de software e os fatores de cada uma delas.

Muitas outras tabelas e questionários existem, também, na literatura que podem ser usados como "checklist" na identificação dos riscos de um projeto. Entre elas destacamos: de Boehm [49], a tabela de itens de riscos e técnicas para gerenciá-los, e de Keane [1], o questionário de taxa de riscos da Harvard Business Review, sua versão para a tabela de Walston e Felix de variáveis que impactam na produtividade, e seu questionário.

Fairley [47] propõe um programa de gerência de riscos para projetos de software, que consiste em:

- se estabelecer uma infraestrutura de projeto, através da qual os fatores de risco podem ser identificados, avaliados, rastreados e solucionados a tempo e de maneira sistemática, e,
- técnicas específicas para atender às tarefas envolvidas numa gerência de riscos sistemática.

A infraestrutura de projeto proposta pelo mesmo autor para a gerência de riscos é composta por:

- padrões e roteiros;
- políticas e procedimentos;
- modelo de processo de desenvolvimento orientado a riscos;
- especificações dos requisitos;
- plano do projeto;
- plano da gerência de riscos;

- garantia da qualidade;
- gerência da configuração;
- plano de documentação, e;
- mecanismos rastreáveis.

A tabela abaixo contém técnicas sugeridas para atender as tarefas da gerência de riscos:

PROCEDIMENTO	TÉCNICA
Identificação dos riscos	cenários de pior caso Análise de Kepner-Tregoe checklists de áreas de risco diagramas de causa-efeito análise dos atributos de custo análise dos cronogramas avaliação da capacidade prototipação
Análise do impacto	Análise Pareto árvores de decisão análise de custo-benefício análises de sensibilidade análise de fatores de qualidade análise da prioridade dos riscos análise de riscos compostos
Preparo do plano de contingência	evitar os riscos transferência dos riscos planos de monitoração dos riscos planos de ação
Monitoração dos riscos	revisões auditorias de projeto métricas de rastreamento monitoração do trabalho rastreamento dos valores gastos
Gerência da recuperação	execução plano de contingência técnicas de gerência de crises repreparo cronogramas/orçamento

Fonte: Fairley [47]

Outra abordagem sobre gerência de riscos que destacamos é a Teoria W proposta por Boehm [55, 58]). A teoria W é uma metodologia fundamentada no princípio de tornar ganhadores todos os envolvidos na realização de um projeto de software, evitando

os conflitos inerentes que existem entre os desejos de cada um.

Além deste princípio fundamental, a Teoria W se apoia também em mais dois princípios subsidiários: "Planejar e Executar o Plano" e "Identificar e Gerenciar os Riscos".

Na Teoria W o gerente é visto como um negociador que acondiciona soluções dentro de condições vitoriosas para todas as partes, estabelece metas, monitora o progresso e procura situações "ganho-perda" e "perda-perda" para transformá-las em situações "ganho-ganho".

A teoria W estabelece um conjunto de precondições "ganho-ganho" para estruturar o processo e o produto resultante de software, apresentadas na tabela I de Boehm [55], e diretrizes estratégicas derivadas destas condições, que se encontram relacionadas nas tabelas II e III do mesmo trabalho.

Os dois princípios subsidiários da teoria W são derivados da combinação de subetapas da estruturação de um processo "ganho-ganho".

Planejar, na teoria W, significa estabelecer um plano de processo que registre o compromisso mútuo entre participantes e forneça uma base para detectar desvios das condições "ganho-ganho" que requerem ações corretivas. Vários tipos de plano estão envolvidos: operacionais, de instalação e treinamento, de suporte ao ciclo de vida e de desenvolvimento. Para cada tipo de plano pode haver planos subsidiários como: planos de gerência da configuração, planos de garantia da qualidade, planos de teste e planos de conversão.

Cada plano da Teoria W é organizado segundo o esquema padrão refletindo questões universais (porque, o que, quando, quem,

onde, como e quanto):

- 1) Objetivos
- 2) Produtos e marcos
- 3) Responsabilidades
- 4) Abordagem
- 5) Recursos

Executar o plano significa gerenciar o projeto aplicando o plano desenvolvido para tornar os participantes vencedores. Isto envolve fazer um esforço particular para monitorar o progresso do projeto em relação ao plano. A natureza deste esforço deve ser especificada no plano e este deve ser revisto e modificado se houver atrasos ou falhas, bem como medidas corretivas devem ser tomadas.

A gerência de riscos da teoria W tem como objetivo assegurar que os planos sejam realísticos concentrando-se nas partes do projeto mais prováveis de ocorrerem problemas que comprometam as condições de ganho dos participantes. Para isto, se divide em duas etapas primárias: a avaliação dos riscos e o tratamento dos riscos. A avaliação dos riscos compreende a identificação dos riscos, a análise dos riscos e a priorização dos riscos. O tratamento dos riscos envolve o planejamento da gerência de riscos, a execução da gerência de riscos e a monitoração e controle dos riscos. Para cada subetapa Boehm [55] sugere técnicas adequadas que podem ser empregadas.

II.2.1.7) Aspectos relacionados ao planejamento da documentação

O planejamento da documentação define os componentes do Plano da Documentação: os guias de estilo para os documentos a

serem produzidos ao longo de todo o ciclo de vida do sistema e os procedimentos para sua publicação.

Os guias de estilo dos documentos são negociados com os envolvidos no projeto e asseguram que (Evans [6]):

- o conteúdo específico de um documento seja pré-definido, garantindo que o documento final atenderá aos padrões e às necessidades do projeto;
- o repositório do documento seja avaliado e esteja de acordo com o que será incluído no documento, antes que qualquer trabalho de desenvolvimento do texto seja feito;
- o desenvolvedor do documento saiba o que deve ser incluído em cada bloco de documento e como integrar esses blocos em uma seção consistente com os requisitos de documentação aprovados.

Baseado nos requisitos do guia de estilo são definidas ferramentas, técnicas, metodologias e requisitos de conteúdo específicos a serem aplicados no desenvolvimento e avaliação de cada componente individual dos documentos. Os requisitos de conteúdo específicos estabelecem o ambiente necessário para produzir os elementos individuais requeridos para cada documento do projeto. A produção do material e o controle do processo deve estar de acordo com as práticas do programa de gerência e controle da configuração do software (Evans [6]).

O plano de documentação deve enfatizar que os documentos correspondentes a cada fase devem ser gerados durante a fase e não após sua conclusão. Para cada um dos documentos deve ser definido no plano (Rocha [2]):

- título;
- escopo;

- um sumário dando o escopo por seção;
- a data que será iniciada e concluída a primeira versão e a data de conclusão da versão final;
- os participantes do aceite da primeira versão e da versão final;
- o tamanho esperado;
- o destinatário de cada documento, e;
- as necessidades de edição.

Os procedimentos para publicação envolvem (Rocha [2]):

- preparação, ou redação original;
- aceite do original;
- forma de redação e edição;
- reprodução (limitada ou em massa), e;
- distribuição (membros do projeto, usuários, terceiros, gerência, arquivamento, etc).

A solução dos problemas de documentação inadequada ou inexistente só pode ser alcançada a partir da conscientização gerencial e do estabelecimento de padrões e regras de documentação. Além disto, esses padrões e regras devem ser revistos e ajustados quando necessário para melhorar a comunicação e assegurar o nível de qualidade.

Em Vale [15] é proposta uma sistemática para o processo de documentação com roteiros de documentos, da qual destacamos as seguintes normas:

- Os documentos devem ser elaborados de acordo com um plano de documentação onde deverão estar definidos padrões de conteúdo, formato e qualidade;

- O responsável por elaborar o plano de documentação deve ser um especialista ou ter bom conhecimento do assunto;

- O público alvo deve ser identificado e seu nível de conhecimento sobre o tema bem como suas limitações devem ser consideradas, para que se possa instruir os documentadores em como redigir os diferentes itens dos documentos de forma clara, concisa e acessível.

II.2.1.8) Aspectos relacionados ao planejamento da gerência da configuração (Evans [6], Koblitz [14])

A gerência e controle da configuração do software começa com o início do projeto e estende-se até a retirada do software de operação. As funções do processo de gerenciamento da configuração do software são:

- Identificação da configuração do software através da rotulação das partes do sistema para visualizá-las;

- Controle da configuração do software, que é a gerência das alterações das partes identificadas do sistema;

- Auditoria da configuração do software, que fornece o mecanismo para verificar o estado corrente do sistema em relação ao estipulado formalmente, e;

- Contabilidade da situação da configuração do software, que fornece o mecanismo para acompanhar e registrar a evolução do sistema.

A gerência da configuração do software garante que os produtos aprovados não sejam violados por alterações descontroladas e indisciplinadas, além de assegurar que a rastreabilidade seja mantida durante o desenvolvimento e que todos os segmentos da infra-estrutura do projeto usem dados

atuais, válidos e controlados.

Toda informação produzida como parte do processo de engenharia de software torna-se parte da configuração do software, e como tal deverá ser organizada de maneira que permita um controle ordenado das alterações. Este controle consiste da identificação e controle das alterações, monitoração para que as alterações sejam implementadas adequadamente e divulgação das alterações.

A gerência da configuração do software está separada em dois segmentos, similares na função e responsabilidades: baseline e projeto. São eles:

1. **Baseline:** A gerência e controle de baselines assegura que as versões formalmente aprovadas e liberadas de toda documentação do software, do hardware e dos resultados de testes permaneçam controladas e que a aplicação seja rastreável em relação aos requisitos.
2. **Projeto:** Controle dos requisitos, do projeto do sistema, do código, e dos parâmetros de teste aprovados, não necessariamente pelos usuários finais do sistema.

O Conselho de Controle da Configuração é o órgão responsável por processar formalmente as alterações propostas para estabelecer baselines. Ele revê, avalia, aprova e libera as principais alterações técnicas e não técnicas, dos componentes do sistema, do hardware e software, e de todas as especificações da baseline.

Todas as alterações requerem revisão e aceitação do usuário final. O controle formal de alterações é o meio pelo qual a correspondência entre as expectativas do usuário final e a

realidade dos desenvolvedores é mantida. O processo de controle de alterações da baseline pode ser dividido em iniciação da requisição de alteração, revisão da requisição de alteração e implementação das alterações aprovadas.

O programa de suporte de biblioteca é o mecanismo administrativo que controla as baselines e processa as alterações. Ele mantém as baselines estabelecidas de projeto, e monitora e controla o desenvolvimento das baselines de projeto e os produtos.

A baseline até ser estabelecida é um conjunto de documentos, materiais ou equipamentos ainda não auditados, em auditoria, reprovados ou aguardando decisão de uma autoridade. Define-se no Plano do Projeto uma sucessão de baselines que coincidirão com os marcos do projeto. As baselines são refinadas até serem estabelecidas quando um marco é alcançado.

Para estabelecer o gerenciamento da configuração são necessárias uma análise detalhada dos requisitos de informações dos usuários finais, ter um banco de dados devido ao grande volume de dados utilizados, formar uma equipe de gerenciamento da configuração e preparar um Plano de Gerenciamento da Configuração do Software (PGCS), estabelecendo as diretrizes, procedimentos e ferramentas a serem utilizadas. O uso de uma ferramenta automatizada de gerenciamento da configuração também deve ser considerado.

A preparação do PGCS deve levar em conta que o mesmo pode evoluir com o sistema, não devendo, portanto, ser estático nem rígido. Um modelo padrão de PGCS aplicável a softwares críticos foi proposto pelo IEEE (IEEE [70]) e sua aplicação, ou uma

adaptação nos casos de softwares não críticos ou já desenvolvidos, é recomendada.

II.2.1.9) Aspectos relacionados ao planejamento do controle da qualidade

A garantia da qualidade do software fornece os mecanismos para incorporar práticas de qualidade ao processo de desenvolvimento do software. Ela envolve tarefas associadas com sete atividades principais (ANSI/IEEE std 730-1981):

- aplicação de métodos e ferramentas ao processo de desenvolvimento;
- condução de revisões técnicas formais;
- teste de software;
- adoção de padrões;
- controle de alterações;
- medição, e;
- manutenção de registros e divulgação.

Teste, verificação e validação fornecem técnicas e ferramentas de suporte aos programas de garantia da qualidade do software. Testes incluem os meios técnicos para verificação, que consiste em garantir que os produtos de uma fase do desenvolvimento preenchem os requisitos estabelecidos na fase anterior, e auxílio à validação, que consiste em se certificar que a especificação dos requisitos reflete apropriadamente o problema em questão. (Cronhjort [33])

O nível de qualidade exigido depende do tipo da aplicação. Porém, qualquer sistema deve atingir um nível satisfatório de qualidade. Para que este nível seja atingido ele precisa ser claramente definido, assim como a maneira como será verificado o

grau com que foram atingidos.

Para isto, alguns aspectos devem ser considerados (Keane [1]):

- O número de walkthroughs ou revisões com o usuário final e o grau de detalhamento necessário dependem: da importância estratégica do projeto para a organização, da estrutura do projeto, do grau de envolvimento do usuário final, e da fase específica que está sendo verificada;
- A intensidade com que os procedimentos de verificação serão usados depende do tamanho e da complexidade do projeto, já que estes dificultam a equipe entender o nível de esforço técnico requerido, principalmente se seus membros não têm experiência em projetos de tamanho e complexidade similar;
- Independente das ferramentas de desenvolvimento usadas no projeto, antes do início da implementação deve ser assegurado que a equipe do projeto e os usuários finais concordam com o escopo e objetivos do projeto, e com os critérios de aceitação do sistema. Além disto, antes de qualquer fase iniciar, os mesmos devem concordar com as tarefas que devem ser executadas e com os produtos que devem ser gerados nesta fase;
- Ao término de cada fase ou a intervalos menores, determinados pelos marcos, deve ser obtida a assinatura do usuário final nos documentos correspondentes àquela fase. O responsável pela aceitação de cada fase deve ser o gerente dos usuários finais que tem a autoridade final sobre o projeto;
- A verificação da fase N de um projeto deve ser feita por participantes das fases N-1 e N+1, e para conduzi-las deve ser escalada uma pessoa experiente e de alto nível;

- Para facilitar o controle gerencial da aceitação dos produtos pode ser usado como instrumento o Cronograma de Entrega de Produtos, que permite registrar o status da aceitação de cada produto, saber o nome do usuário final ou gerente responsável pela aceitação e saber que produtos foram entregues dentro do prazo ou não, e quais ainda devem ser entregues;
- Para determinar a frequência de aceite pode-se usar uma regra: planejar a aceitação a intervalos de no máximo o percentual correspondente à taxa de tempo não produtivo da organização. Assim, se o tempo produtivo dos funcionários de uma organização é 80% do tempo total, não mais que 20% do trabalho total do projeto pode ser executado sem que seja aprovado. Dessa forma, em caso de necessidade, o gerente do projeto pode forçar o pessoal a ter 100% de seu tempo voltados para o projeto.

O controle da qualidade deve ser planejado e formalizado num plano específico. Para isto, pode-se adaptar um plano padrão do órgão responsável pelo controle da qualidade na organização às características e necessidades particulares do projeto. Na falta deste pode ser adaptado o modelo padrão de plano do controle da qualidade para softwares críticos do IEEE (IEEE [71]).

II.2.2) ACOMPANHAMENTO DO PROJETO

Segundo Keane [1], um dos maiores problemas do processo de desenvolvimento é a dificuldade em fazer o acompanhamento do progresso de forma efetiva que permita saber o status do projeto num dado momento, e determinar a causa de possíveis problemas. Este problema é consequência da subjetividade da maioria dos métodos de relatar o trabalho executado, que freqüentemente levam

a síndrome dos 90%, criam um falso otimismo e falham na detecção de problemas em tempo hábil para solucioná-los.

Para eliminar esses problemas é necessário um método que meça resultados tangíveis e permita fazer um acompanhamento preciso do desenvolvimento do projeto. Qualquer aspecto do projeto que precise de controle também precisa de medição, dado que não se pode controlar o que não se pode medir. A extensão do controle vai depender da precisão da medição, e qualquer coisa que não se meça estará fora de controle (DeMarco [5]).

Além disto, também para assegurar a disponibilidade de informações históricas, é necessário instituir uma política meticulosa de medição e documentação durante o projeto. Toda organização para ser capaz de aplicar os princípios gerenciais que guiam o planejamento, a tomada de decisão e a monitoração de futuros projetos deve definir e adotar métricas, coletar medidas de projetos existentes e validar essas métricas. Na ausência de métricas, não existem meios para medir se está ocorrendo progresso e, se estiver, à que taxa (Ghezzi [17]) (DeMarco [5]).

Conforme discutimos na seção referente ao planejamento do acompanhamento, as ferramentas básicas para fazer o acompanhamento e controlar o projeto consistem de ferramentas de programação e controle de atividades e de planilhas e gráficos para planejamento da distribuição das atividades, do orçamento e dos recursos, e para registro da evolução do progresso e do consumo de recursos durante o projeto.

Todo acompanhamento deverá ter como referência básica o Plano do Projeto. Enquanto o plano descreve o que deve acontecer,

os relatórios de progresso medem o que realmente aconteceu em relação ao planejado, devendo responder à questões como (Keane [1]):

- Que produtos já foram iniciados, completados, aceitos?
- Que variações existem no trabalho concluído até a data?
- Que variações potenciais existem no trabalho futuro?
- Que problemas existem que podem afetar a execução do plano? O que foi ou o que pode ser feito para resolvê-los?
- Qual é o tempo e custo estimado para completar o projeto?

O gerente do projeto não deve confiar, porém, apenas em relatórios de acompanhamento e deve estar alerta para outros sinais que evidenciam problemas, como (Keane [1]):

- desvios do plano;
- relatórios se tornam subjetivos ou cessam;
- o projeto começa a perder o rumo;
- as pessoas da equipe não se mostram decididas e positivas;
- aparecem acusações de falta de cooperação entre os membros da equipe;
- a conformidade com o cronograma é perdida;
- a motivação do grupo se torna preocupante;
- a comunicação diminui, e;
- atalhos para encurtar as tarefas começam a ser usados.

Independentemente de como os desvios ao plano são detectados, o gerente do projeto deve decidir como tratá-los. Segundo Keane [1], um atraso é sério a ponto de comprometer o prazo de entrega do projeto se a estimativa para concluir as tarefas atrasadas ultrapassa o orçamento originalmente designado para elas num percentual superior a taxa de tempo não produtivo

da organização. Esta regra se baseia no fato que em caso de necessidade se pode forçar a equipe a dedicar 100% do seu tempo ao projeto.

As opções para tratar os desvios são limitadas mas as alternativas devem ser consideradas cuidadosamente pelo gerente do projeto. Ghezzi [17] considera como opções:

- Alocar temporariamente profissionais "senior" ou contratar consultores especialistas para as partes do projeto que estão sofrendo problemas;
- Examinar os requisitos e remover aqueles que não são absolutamente necessários, ou seja, aqueles que apenas têm como objetivo dar melhor aparência mas não acrescentam nada substancial ao produto.

Já em Keane [1] são sugeridas, entre outras, as seguintes diretrizes:

- Fazer uma avaliação objetiva da situação do projeto;
- Estabelecer um plano de contingência;
- Comunicar aos usuários finais e gerência a extensão do problema, incluindo um novo cronograma se necessário;
- Apresentar a possibilidade de redefinir o projeto para mantê-lo dentro do orçamento;
- Estabelecer e insistir em revisões mais freqüentes, e em comunicações ou reuniões até diárias, se for o caso;
- Reafirmar as responsabilidades do usuários finais, critérios de aceitação e procedimentos de alterações;
- Como medida preventiva, rever o plano do projeto após a primeira semana ou consumidos 10 a 15% do orçamento, para verificar se é necessário alterar os intervalos entre os marcos.

Esta última diretriz propõe uma revisão a esta altura porque após aproximadamente a primeira semana muitas indefinições se esclarecem, já se pode identificar se a equipe corresponde ao previsto e se o usuário final tem o conhecimento esperado sobre a aplicação, permitindo, assim, tomar providências à tempo.

O uso de auditorias é indicado, também, por Keane [1] como meio efetivo para reduzir o número de atrasos. O auditor deve examinar o plano do projeto para determinar a situação do projeto e verificar que produtos estão no prazo, quais estão atrasados e quais foram concluídos com gastos e tempo maiores que os planejados. Para determinar a causa dos problemas deve ser examinado o livro de controle do projeto, que contém todos os relatórios semanais, de conclusão das tarefas e de alterações.

Os seguintes critérios podem ser usados pelo gerente do projeto para determinar se o projeto precisa ser auditado (Keane [1]):

- se o projeto já tem um excedente de mais de 2.500 horas;
- se o projeto envolve mais de 5 pessoas;
- se o projeto envolve uma grande soma em dinheiro a ser gasta num curto período de tempo;
- se o projeto for especialmente preocupante para o gerente, independente de seu custo;
- quando o prazo final é crítico.

No caso de não se encontrar solução apropriada para recuperar um atraso, a melhor alternativa é reconhecê-lo e revisar o Plano do Projeto para adequá-lo às novas informações que se tem sobre a dificuldade das tarefas, capacidade da equipe

e disponibilidade de recursos (Ghezzi [17]).

Durante todo o acompanhamento o plano do projeto deve ser mantido atualizado e todas as suas versões devem se tornar parte de um livro de controle do gerente do projeto, para permitir que se possa rastrear tudo que ocorreu durante o projeto. Se após uma revisão for identificada uma necessidade de alterar o Plano do Projeto, mesmo que esta não altere o custo ou prazo do projeto, deve-se iniciar um procedimento de alteração e obter formalmente a aprovação do usuário final.

II.3) MÉTODOS DE GERÊNCIA DE PROJETOS

Métodos de apoio ao desenvolvimento de software são definidos em Rocha [53] como "conjuntos de diretivas para a seleção e aplicação de técnicas e instrumentos, de maneira a organizar o pensamento e o trabalho do usuário (engenheiro de sw) ao longo do processo de desenvolvimento de software".

Os métodos fornecem as técnicas, instrumentos e ferramentas que viabilizam a realização sistemática das atividades definidas no ciclo de vida de um software. As técnicas são conjuntos de princípios para a execução de uma tarefa específica do processo de desenvolvimento. Os instrumentos possibilitam a utilização de um método e as ferramentas permitem utilizar os instrumentos de forma mais produtiva. (Aguiar [23])

As técnicas gerenciais oferecem os meios para planejar e controlar o processo de desenvolvimento de um software, enquanto que os instrumentos de apoio à gerência viabilizam a utilização das mesmas, tornando possível este planejamento e controle. (Aguiar [23])

A partir da pesquisa realizada, selecionamos para descrever

nesta seção o método de gerência de projetos de software de Abdel-Hamid (Abdel-Hamid [59] Abdel-Hamid [60]) e alguns métodos de apoio a determinadas atividades específicas de planejamento de projetos.

II.3.1) MODELO INTEGRATIVO E DINÂMICO DE GERÊNCIA DE PROJETOS DE SOFTWARE (ABDEL-HAMID [59,60])

O modelo integrativo e dinâmico de gerência de projetos de software tem como base dois pontos:

1) Uma perspectiva integrada baseada no fato do sistema de gerência de software ser um conglomerado de funções interdependentes e interrelacionadas, conforme é ilustrado na figura abaixo, mostrando várias variáveis tangíveis e intangíveis que têm impacto no processo de desenvolvimento do software.

Fig. II.1: Sistema de Gerência de Software.

Fonte: Abdel-Hamid [59].

2) Processos realimentadores aplicados à gerência de projetos de software permitindo entender o comportamento de tais projetos.

O modelo proposto foi constituído a partir de estudos realizados em cinco organizações de software e a partir de estudo da literatura, e tem a seguinte estrutura, apresentada na figura a seguir: (figura II.2)

Fig. II.2: Estrutura do Modelo.

Fonte: Abdel-Hamid [59]

Além dos modelos completos de gerência de projetos, muitos outros modelos existem para apoiar o gerente de projetos em determinados aspectos da gerência. A maioria desses modelos é

voltada para etapas do planejamento e objetivam principalmente a estimativa de recursos.

A seguir discutiremos sobre esses modelos de estimativa de recursos, apresentando suas principais características gerais, as características e deficiências particulares dos três principais métodos existentes de estimativa de recursos, bem como algumas extensões a esses modelos, e concluímos analisando a questão da escolha do método mais adequado a um projeto e os avanços que têm surgido para suprir deficiências gerais desses métodos.

II.3.2) MODELOS DE ESTIMATIVA DE RECURSOS

Hoje dispomos de vários métodos para estimar recursos. Estes métodos ainda hoje podem ser classificados, conforme Boehm [4], em:

- a) **Método de estimativa por analogia**, onde os custos são estimados a partir do estudo de projetos passados similares;
- b) **Métodos "top-down"**, onde as estimativas são baseadas na análise das propriedades globais do software, que depois são divididas pelos componentes do produto.
- c) **Métodos "bottom-up"**, onde os custos dos componentes são estimados geralmente pela pessoa que será responsável por seu desenvolvimento, e depois são somados para se obter o custo total.
- d) **Métodos baseados em julgamento de especialistas**, que consistem de consultas a especialistas que usarão suas experiências profissionais e conhecimento do projeto a ser desenvolvido para estimar seu custo, através de técnicas como a de Delphi (Helmer [40]) ou a de Wideband Delphi (Boehm [4]).

e) **Modelos algorítmicos**, que são compostos por um ou mais algoritmos matemáticos que calculam as estimativas em função de variáveis que influem nos custos. Dependendo da forma de seus algoritmos, esses modelos podem ser classificados em lineares, multiplicativos, analíticos, tabulares ou compostos, conforme Boehm [4], ou ainda, como modelos estáticos de variável única, modelos estáticos multivariáveis ou modelos dinâmicos multivariáveis, segundo Basili [41].

A maioria dos modelos existentes para estimar recursos é algorítmico, compostos de fórmulas derivadas empiricamente, e baseiam a estimativa de esforço no tamanho do código.

Além de basearem a estimativa de esforço no tamanho do código, a maioria desses modelos também compartilha certos conceitos. O esforço total de desenvolvimento requerido para produzir um software ou parte dele é estimado em termos do número de pessoas e do tempo necessários para desenvolvê-lo. Para contabilizar as muitas variáveis que podem afetar a produtividade do projeto, atributos de custo são usados para ajustar esta estimativa inicial. Em geral, os atributos de custo podem ser classificados como sendo atributos do produto, do computador, do pessoal ou do projeto, e variam nestas classes entre os modelos (Ghezzi [17]).

A tabela a seguir mostra um conjunto de atributos considerados em vários modelos, classificados em cinco diferentes grupos.

GRUPO	FATOR
Atributos de tamanho	Instruções fonte Instruções objeto Número de rotinas Número de itens de dados Número de formatos de saída Documentação Número de pessoal
Atributos do programa	Tipo Complexidade Linguagem Reutilização Confiabilidade requerida Requisitos de exibição
Atributos do computador	Restrições de tempo Restrições de memória Configuração do hardware Desenvolvimento concorrente de HW SW de interface com o equipamento
Atributos de pessoal	Capacidade do pessoal Continuidade do pessoal Experiência com o hardware Experiência com a aplicação Experiência com a linguagem
Atributos de projeto	Ferramentas e técnicas Interface com o cliente Definição dos requisitos Volatilidade dos requisitos Cronograma Segurança Acesso ao computador Viagens/relocações/múltiplos locais Maturidade do suporte

Tab. II.1: Fatores usados em modelos de estimativa de custo.

Fonte: Ghezzi [17].

As etapas básicas para estimar os custos através destes modelos são as seguintes: (Ghezzi [17])

- 1 - Estima-se o tamanho do software e aplica-se a fórmula do modelo para obter uma estimativa inicial.
- 2 - Ajusta-se a estimativa através dos atributos de custo ou

outros fatores de escala do modelo.

3 - Aplica-se as ferramentas do modelo à estimativa obtida em (2) para determinar o esforço total, distribuição por atividade, etc.

4 - Validam-se os resultados contra uma base de dados de projetos da organização. Esta base pode ser usada para validar a estimativa de um determinado modelo contra projetos passados, para calibrar atributos de custo ou fatores de ajuste do modelo ao ambiente particular da organização, para fazer uma estimativa inicial de tamanho, ou para calibrar estimativas de esforço derivadas de um modelo.

Os modelos mais populares de estimativa de custos, segundo Navlakha [36], são: o modelo COCOMO de Boehm, o modelo SLIM de Putnam, o modelo RCA PRICES, o modelo SEER de Jensen e o modelo SOFCOST de Grumman, todos comercialmente disponíveis e sendo usados por muitas organizações de software. Ele inclui ainda o modelo Wolverton da IBM, o modelo Doty, o modelo SCD e o modelo COPMO. Além destes, destacamos ainda as técnicas analíticas, como a análise dos Pontos por Função (Albrecht [32]), e o Sistema BANG (DeMarco [5, 42]).

A seguir apresentamos três destes modelos, que consideramos como os mais importantes: o modelo COCOMO, Pontos por Função e o modelo de Putnam. São apresentadas, também, uma extensão ao modelo COCOMO proposta por Gulezian [43] e uma extensão ao método Pontos por Função, o modelo MACOS de Guerra [39].

II.3.2.1) MODELO COCOMO (Boehm [4])

Foi desenvolvido por Boehm com o objetivo de calcular o esforço e tempo de desenvolvimento de um projeto de software. É o

modelo algorítmico mais completo e pode ser aplicado a projetos de variados tipos e tamanhos devido a sua flexibilidade. Possui três formas progressivamente mais detalhadas e precisas: COCOMO Básico, COCOMO Intermediário e COCOMO Detalhado.

a1) COCOMO BÁSICO

Consiste de uma simples fórmula de estimativa de custo em função apenas do tamanho do software em número de instruções fontes entregues, sendo de precisão limitada porque não considera fatores que têm influência nos custos. É útil para estimativas rápidas logo no início do projeto, seja de desenvolvimento ou de manutenção.

O esforço e tempo de desenvolvimento são calculados de acordo com o nível de complexidade do sistema, a partir da estimativa do tamanho, conforme a tabela abaixo:

NÍVEL DE COMPLEXIDADE	ESFORÇO	TEMPO
ORGÂNICO	$MM = 2,4(KDSI)^{1,05}$	$TDEV = 2,5(MM)^{0,38}$
DIFUSO	$MM = 3,0(KDSI)^{1,12}$	$TDEV = 2,5(MM)^{0,35}$
RESTRITO	$MM = 3,6(KDSI)^{1,20}$	$TDEV = 2,5(MM)^{0,32}$

Tab. II.2: Equações de esforço e tempo para o COCOMO Básico.

Fonte: Boehm [4]

Na tabela MM é o número de homens-mês ("number of man-months") e KDSI são milhares de instruções fontes entregues ("Thousands of delivered source instructions").

O modelo permite, ainda, estimar a produtividade (KDSI/MM) e a equipe média (MM/TDEV), e fazer a distribuição de esforço e tempo por fases e por fases e atividades, de acordo com a

complexidade e tamanho do software. Para fazer a distribuição o modelo fornece tabelas de percentual de distribuição por fases e fases e atividades (tabelas 6-8, 7-1, 7-2 e 7-3 de Boehm [4]).

A estimativa do esforço por ano de manutenção é feita em termos da variável ACT (Annual Change Traffic), que representa a fração de instruções fonte do produto que sofrerá alterações durante um ano, seja por adição ou por modificação. O esforço anual com manutenção em homens-mês é estimado a partir do esforço estimado de desenvolvimento, e o número de pessoas em tempo integral requeridas para manutenção, considerando uma pessoa dedicada em tempo integral que contribui com 12 homens-meses num ano, é obtido dividindo-se o esforço anual com manutenção por 12.

a2) COCOMO INTERMEDIÁRIO

Neste modelo foram ajustadas as equações de esforço e incluídos atributos que influenciam nos custos, que transformados num fator de multiplicação que representa o efeito do atributo no esforço de desenvolvimento, permitem refinar as estimativas. Este modelo permite, ainda, fazer análises de sensibilidade com respeito aos atributos, estimando-se o efeito no custo de desenvolvimento causado por mudanças no comportamento desses atributos.

Além da estimativa geral, adequada apenas para estágios iniciais do projeto, é oferecida também a opção de se estimar o esforço para cada componente do software, o que gera um resultado mais preciso, já que permite refletir nos custos diferenças no comportamento dos atributos entre os diversos componentes.

As equações ajustadas de cálculo do esforço são as seguintes: (Tabela II.3)

NÍVEL DE COMPLEXIDADE	ESFORÇO
ORGÂNICO	$MM = 3,2(KDSI)^{1,05}$
DIFUSO	$MM = 3,0(KDSI)^{1,12}$
RESTRITO	$MM = 2,8(KDSI)^{1,20}$

Tab.II.3: Equações de esforço e tempo do COCOMO Intermediário.

Fonte: Boehm [4]

Os quinze multiplicadores para ajuste do esforço calculado, cujos valores constam da tabela 8-2 de Boehm [4] e são explicados nas tabelas 8-3 e 8-4 de taxa de influência dos fatores de custo do modelo, são:

- Atributos do Produto: integridade, tamanho do banco de dados e complexidade;
- Atributos do computador: tempo de execução, memória principal, volatilidade da máquina virtual e tempo de resposta;
- Atributos de pessoal: capacidade do analista, experiência com a aplicação, capacidade do programador, experiência com a máquina virtual e experiência com a linguagem;
- Atributos do projeto: práticas modernas de programação, uso de ferramentas de software e tempo de desenvolvimento desejado.

O tempo estimado de desenvolvimento, TDEV, é calculado através das mesmas equações do COCOMO Básico e as distribuições de esforço e tempo por fases e por fases e atividades são obtidas pelas respectivas tabelas do COCOMO Básico.

Para calcular o esforço na fase de manutenção pode-se usar

para os atributos de custo os mesmos multiplicadores da fase de desenvolvimento, ajustando apenas os que variam nesta fase. As linhas correspondentes modificadas da tabela de multiplicadores de esforço constam das tabelas 8-7 e 8-8 de Boehm [4].

Para estimativas a partir dos componentes do produto, o esforço de desenvolvimento, (MM_{nom}) , deve ser calculado somando-se os tamanhos estimados dos componentes em DSI e aplicando-se a equação de cálculo do esforço correspondente ao nível de complexidade do software.

Calculando-se a produtividade, $(DSI / MM)_{nom} = (total\ DSI) / MM_{nom}$, pode-se, ainda, obter o esforço para cada componente, $(MM_{nom} = DSI / (DSI / MM)_{nom})$, aos quais são então aplicados os multiplicadores de esforço para ajustá-los. A produtividade por componente (DSI/MM) é estimada a partir do esforço por componente.

a3) COCOMO DETALHADO

Este modelo supre as principais limitações do modelo intermediário permitindo contabilizar a influência dos atributos de custo em cada fase do projeto, e tratando cada atributo dentro do nível hierárquico que tem influência, módulo, subsistema ou sistema.

O nível modular tem como parâmetros o número de instruções fonte entregues e os atributos de custo que variam entre os módulos: complexidade, capacidade dos programadores, experiência dos programadores com a máquina virtual e com a linguagem de programação.

Os subsistemas são descritos pelos demais atributos que tendem a variar entre subsistemas mas tendem a permanecer iguais

para todos os seus módulos.

O nível de sistemas é usado para aplicar relações de projeto como um todo, tal como as equações de esforço e tempo por fase.

Para cada atributo de custo o modelo tem um conjunto de tabelas que dão para cada nível de influência do atributo um valor de multiplicador de esforço separado contabilizando seu reflexo nas principais fases do desenvolvimento.

Os procedimento para calcular o esforço por módulo, subsistema e sistema são os seguintes:

1. O esforço para desenvolver o sistema, $(MM)_{nom}$, e a produtividade, $(DSI/MM)_{nom}$, são obtidos através das equações do COCOMO Intermediário a partir da soma do tamanho de cada subsistema, obtidos, por sua vez, a partir da soma do tamanho estimado de seus módulo, em DSI.
2. A distribuição do esforço por fase do desenvolvimento, $(FRAC)_p$, é obtida através da tabela de distribuição do esforço por fase do COCOMO Básico.
3. A distribuição do esforço por fase para cada módulo é calculada como:

$$(MM)_{nom.p} = [(DSI \cdot (FRAC)_p] / [(DSI / MM)_{nom}].$$

4. A distribuição do esforço por fase para cada módulo é ajustada através dos multiplicadores de esforço por fase para o nível modular apresentados na tabela 23-2 de Boehm [4].
5. O esforço por subsistema, $(MM)_{mod.p}$, é obtido somando por fase os esforços de seus módulos componentes.
6. Para cada subsistema e cada fase, a soma dos esforços ajustados de cada módulo multiplicados pelos valores dos atributos à nível de subsistema (tabela 23-3) fornecem o esforço

por fase para cada subsistema, $(MM)_{est.p}$.

7. Para cada fase, a soma dos esforços estimados para todos os subsistemas fornecem o esforço por fase para o sistema. A soma dos esforços de cada fase é o esforço total de desenvolvimento do sistema.

8. O tempo de desenvolvimento é estimado por uma das equações do COCOMO Básico, de acordo com o nível de complexidade do software.

9. O cálculo do esforço com manutenção e a distribuição do esforço por atividades são feitos da mesma maneira que para o COCOMO Intermediário.

O modelo COCOMO, apesar de ser o mais reconhecido e utilizado, apresenta deficiências. As principais deficiências apontadas são (Conte [31], Gulezian [43]):

- sua variável de entrada, linhas de código, não possibilita sua utilização de forma confiável em estágios iniciais do desenvolvimento;
- O modelo se aplica a ciclo de vida em cascata;
- Os valores dos atributos de custo para os projetos de seu banco de dados de origem foram obtidos por julgamento de especialistas, o que os torna sujeitos a erros, especialmente no caso dos projetos mais antigos e sem uma documentação completa;
- As equações derivadas no COCOMO não foram obtidas diretamente de um método de regressão de mínimos quadrados;
- O desempenho do COCOMO Básico não é bom quando aplicado a seu próprio banco de dados;
- O grande número de escolhas para os parâmetros de ajuste fornecem ao modelo grande flexibilidade, ao mesmo tempo que lhe dão grande volatilidade, requerendo ao gerente de projeto suprir

uma grande quantidade de informações para obter o esforço estimado;

- Desde que os parâmetros de entrada do modelo foram desenvolvidos heurísticamente, transportar o modelo em sua forma presente para ambientes diferentes daqueles correspondentes ao de seu desenvolvimento é extremamente difícil. Além disto, o procedimento recomendado para calibrar o modelo para outros ambientes restringe-se aos coeficientes das equações de estimativa, sem considerar os atributos de custo, que permanecem os mesmos. Uma vez que estes atributos de custo são também dependentes do banco de dados da TRW, eles necessitam ser calibrados para novos ambientes.

Para suprir as deficiências do modelo em relação a sua aplicação a ambientes diferentes daquele onde foi desenvolvido, Gulezian [43] propõe uma reformulação estatística do modelo e uma metodologia para calibrá-lo completamente e automaticamente em termos dessa reformulação, que apresentamos a seguir.

Gulezian selecionou para seu trabalho o modelo COCOMO intermediário, considerando os fatos deste ser mais simples que o detalhado e do modelo detalhado não acrescentar precisão significativa aos resultados. A estimativa de esforço do COCOMO intermediário pode ser escrita como:

$$MM_{est} = MM_{nom} * FAE = a(KDSI)^b \prod_{i=1}^{16} C_{ij}, \quad j=1,2,\dots,k_i$$

onde:

FAE = fator de ajuste do esforço;

C_{ij} = multiplicador correspondente a j-ésima categoria selecionada para o i-ésimo atributo de custo, e;

k_i = número de categorias correspondentes ao i-ésimo atributo de custo.

A metodologia para calibrar o modelo de acordo com os objetivos estabelecidos foi obtida reformulando o método COCOMO como uma regressão multivariável, de forma que os seis coeficientes e os setenta e quatro atributos de custo sejam estimados em termos de um conjunto menor de coeficientes estimados simultaneamente. Para isto a seguinte estrutura é usada:

$$MM_{est} = a'a_1'm_1a_3'm_3(KDSI)^b + b_1'm_1 + b_3'm_3 \prod_{i=1}^{16} d_i^{C_{ij}} - 1,$$

$$j=1,2,\dots,k_i$$

onde

a' = coeficiente nominal correspondente ao coeficiente a para utilitários;

a_1' = múltiplo de a' que produz o coeficiente nominal para aplicativos;

a_3' = múltiplo de a' que produz o coeficiente nominal para sistemas operacionais;

b' = coeficiente nominal correspondente a b , para utilitários;

b_1' = adição a b' que produz o coeficiente nominal para aplicativos;

b_3' = adição a b' que produz o coeficiente nominal para sistemas operacionais;

m_1 = variável binária que assume o valor 1 para aplicativos e zero para os demais modos;

m_3 = variável binária que assume o valor 1 para sistemas operacionais e zero para os demais modos;

d_i' = coeficiente correspondente ao i-ésimo atributo de custo.

O valor de um multiplicador calibrado é obtido a partir do valor estimado de d' , através da equação:

$$C_{ij} = d_i' C_{ij} - 1.$$

Os valores dos coeficientes nominais correspondentes aos modos do modelo COCOMO original são obtidos a partir dos coeficientes estimados, através da tabela abaixo:

Modo	Equivalente COCOMO	
	a	b
Orgânico	$a_1' a'$	$b_1' + b'$
Difuso	a'	b'
Restrito	$a_3' a'$	$b_3' + b'$

Tab.II.4: Coeficientes nominais correspondentes ao COCOMO original

Fonte: Boehm [4]

O método de solução proposto é a estimativa por mínimos quadrados tradicional, obtida linearizando a equação (1) através de logaritmo, o que resulta em:

$$\ln MM_{est} = A' + A_1' m_1 + A_3' m_3 + b' \ln KDSI + b_1' (m_1 \ln KDSI) + b_3' (m_3 \ln KDSI) + \sum_{i=1}^{16} d_i' (C_{ij} - 1),$$

onde

$$A' = \ln a';$$

$$A_1' = \ln a_1';$$

$$A_3' = \ln a_3';$$

$$D_i' = \ln d_i'.$$

Os procedimentos seguintes são estimar A' , A_1' , A_3' , b' , b_1' , b_3' e D_i' diretamente minimizando a soma dos quadrados do logaritmo do esforço em homens-meses, e converter os termos apropriados para as unidades originais. A soma dos quadrados a ser minimizada diretamente tem a forma:

$$[\ln MM_{\text{real}} - \ln MM_{\text{est}}]^2.$$

Para demonstrar a efetividade deste procedimento, Gulezian o aplicou ao banco de dados original do COCOMO. Os resultados demonstraram que, de acordo com os critérios determinados, o procedimento é plenamente satisfatório.

As características estatísticas adicionadas, não existentes no modelo COCOMO em sua forma original, fazem a superioridade de sua reformulação:

- A reformulação utiliza valores baseados em dados para os coeficientes, que são todos simultaneamente estimados para levar em conta os interrelacionamentos entre todas as variáveis relevantes;
- Uma medida de erro definida e calculável pode ser amarrada às estimativas dos coeficientes, à equação de estimação, e as previsões baseadas na aplicação da equação;
- O modelo reformulado possui capacidade para estabelecer intervalos probabilísticos de estimativa de esforço;
- Medidas e métodos de acompanhamento estão disponíveis para avaliar a importância individual dos atributos de custos.

Além disto, conforme Gulezian, utilizando a reformulação proposta é possível usar o modelo como ponto de partida para novos esforços de modelagem, modificando o modelo

subsequentemente para obter um modelo verdadeiramente específico para diferentes ambientes. Isto inclui a validação do modelo com novos dados e durante a aplicação do modelo, quando usado para obter estimativas.

II.3.2.2) PONTOS POR FUNÇÃO (Abrecht [32])(Machado [13])

É uma forma alternativa ao cálculo do número de linhas de código fonte, desenvolvida por Allan Albrecht, que avalia o tamanho do software com base nas funções que o mesmo executa. No cálculo não são considerados fatores externos como: experiência da equipe, uso de ferramentas automatizadas, uso de métodos estruturados de desenvolvimento, linguagem de programação, entre outros, porque embora esses fatores sejam responsáveis pelo nível de produtividade da equipe, não determinam o tamanho do software.

Além de medir o tamanho do software baseado nas funções que ele executa, de acordo com a visão do usuário, este modelo propicia:

- estabelecer, com consistência, os custos de desenvolvimento e manutenção de software;
- determinar a produtividade no desenvolvimento e manutenção do software;
- determinar tendências de produtividade e identificar os fatores responsáveis;
- avaliar o impacto na produtividade de medidas na produção e manutenção de software;
- justificar medidas e providências que sejam requeridas para aumentar a produtividade; e
- avaliar conveniências entre expandir ou desenvolver de

novos, entre manutenção ou substituição, e entre compra ou desenvolvimento interno.

A medida do tamanho do software baseada nas suas funções reflete também a expressão do trabalho de desenvolvimento de um software novo, ou de uma expansão, ou da manutenção de software já existente.

A estimativa inicial é gerada calculando-se os pontos por função e dividindo-os pela fração de PF/LOC da linguagem de programação a ser usada no projeto, obtendo-se o tamanho estimado.

A seguir são resumidos os procedimentos de cálculo das estimativas de tamanho do software, do esforço e da produtividade.

i. Procedimentos para determinação do tamanho em pontos por função:

i.1) Identifica-se e conta-se cada um dos principais tipos de dados ou controles existentes na aplicação. A contagem de cada tipo deve ser separada de acordo com três níveis de complexidade (baixa, média ou alta) associada a cada um. Os tipos são:

- a) Entradas externas (EE): tipos de entradas diferentes provenientes do ambiente externo à aplicação;
- b) Saídas externas (SE): tipos de saídas produzidas pela aplicação;
- c) Consultas externas (CE): tipos de consultas diferentes provenientes do ambiente externo à aplicação;
- d) Arquivos lógicos internos (AI): tipos de arquivos (visão lógica do usuário) diferentes existentes na aplicação;
- e) Arquivos de interfaces externas (IE): tipos diferentes de interfaces com tais aplicações.

i.2) Multiplica-se os totais de cada tipo e grau de complexidade pelos fatores de peso associados, conforme abaixo:

$$\begin{aligned} \text{PFB} = & (\text{QT_EEcs} \times 3) + (\text{QT_EEcm} \times 4) + (\text{QT_EEcc} \times 6) + \\ & (\text{QT_SEcs} \times 4) + (\text{QT_SEcm} \times 5) + (\text{QT_SEcc} \times 7) + \\ & (\text{QT_AIcs} \times 7) + (\text{QT_AIcm} \times 10) + (\text{QT_AIcc} \times 15) + \\ & (\text{QT_IEcs} \times 5) + (\text{QT_IEcm} \times 7) + (\text{QT_IEcc} \times 10) + \\ & (\text{QT_CEcs} \times 3) + (\text{QT_CEcm} \times 4) + (\text{QT_CEcc} \times 6) \end{aligned}$$

QT é a quantidade de cada tipo e cs, cm e cc representam, respectivamente, complexidade simples, média e alta.

i.3) Calcula-se o Fator de Ajuste a ser aplicado ao total dos Pontos por Função Brutos, refletindo o grau de influência que as características gerais da aplicação exercem.

A soma dos graus de influência estimados para cada característica, SGI, é aplicada na equação de cálculo do Fator de Ajuste resultando num fator de ajuste entre 0,65 e 1,35, ou seja $\pm 35\%$. O fator é, então, multiplicado pelo total de Pontos por Função Brutos, PFB, para obter-se o tamanho do software em Pontos por Função.

$$\text{PF} = \text{PFB} \times \text{FA}$$

$$\text{FA} = 0,65 + (0,01 \times \text{SGI})$$

As características gerais são: uso de teleprocessamento, processamento distribuído, objetivos de desempenho, restrições de configuração, volume de transação, entrada de dados "online", transações interativas "online", complexidade do processamento, projeto visando reutilizabilidade do código, facilidade de conversão e instalação, facilidade de operação, instalação em múltiplos locais, e facilidade de uso e manutenção.

Os Graus de influência têm a seguinte classificação:

Ausente ou sem influência	= 0
Influência insignificante	= 1
Influência moderada	= 2
Influência média	= 3
Influência significativa	= 4
Influência forte e generalizada	= 5

i.4) Cálculo do produto do trabalho

Os pontos por função representam para software existente o tamanho da aplicação, para desenvolvimento, o tamanho do trabalho necessário, e para manutenção, o tamanho do trabalho necessário para realizar a manutenção/suporte durante o período de um ano.

. A Fórmula geral para cálculo do tamanho ou Produto do Trabalho para software existente é:

$$PFE = PFB \times FAE.$$

. Para Produto do Trabalho de desenvolvimento sua fórmula geral aplica-se tanto para desenvolvimento novo como para expansão de software existente, sendo que apenas os termos PFB_{inc} e FAE_{dep} serão diferentes de zero, no primeiro caso.

$$PFD = (PFB_{inc} + PFB_{alt-dep}) \times FAE_{dep} + (PFB_{exc} \times FAE_{ant})$$

. O Produto do Trabalho de suporte/manutenção pode ser calculado ou recontando-se os Pontos por Função do software como um todo após as mudanças e usando-se o mesmo procedimento do cálculo do Produto do Trabalho para software existente, ou através de sua fórmula geral:

$$PFS = PF_{ant} + (PFB_{inc} + PFB_{alt-dep}) \times FAE_{dep} - \\ (PFB_{exc} + PFB_{alt-ant}) \times FAE_{ant}$$

ii. Deteminação do esforço do trabalho

A determinação do esforço do trabalho tem como objetivo

determinar a produtividade do projeto que deve ser armazenada numa base de informações históricas para viabilizar futuras estimativas. É feita através do registro sistemático de todo o esforço investido em um projeto de desenvolvimento, expansão ou suporte/manutenção, por parte das pessoas que, direta e indiretamente, gastaram seu tempo em atividades ligadas ao projeto. Entre estas pessoas inclui-se pessoal de desenvolvimento e suporte/manutenção, usuários, contratados externos, digitadores ou conferentes ligados ao desenvolvimento (programas, massas de teste, etc), auxiliares administrativos, secretárias, etc.

A apuração do esforço do trabalho deve ser feita nas seguintes ocasiões:

- conclusão de programa: para calcular a produtividade naquele programa, o que permite avaliar a produtividade individual;

- conclusão de rotina ou módulo: para calcular a produtividade naquela rotina ou módulo, o que permite avaliar a equipe responsável por ela;

- conclusão de aplicação: para calcular a produtividade naquela aplicação, o que permite avaliar a equipe responsável por ela;

- conclusão do projeto: para calcular a produtividade naquele projeto, o que permite avaliar a produtividade do projeto ou da equipe responsável por ele;

- encerramento do ano: para calcular a produtividade na manutenção/suporte, o que permite avaliar a equipe ligada a esta atividade. O esforço do trabalho total da instalação é calculado para determinar e avaliar a produtividade do órgão produtor de

software. Neste cálculo entram projetos que foram cancelados, independentemente dos motivos, mas não entram atividades ocorridas durante a fase de manutenção e as de projetos ainda não concluídos.

Observa-se, também, que as somas não são cumulativas porque existem atividades que não são específicas de um programa, de uma rotina ou módulo, de uma aplicação, ou de um projeto.

Uma matriz de fases e atividades deve ser preenchida com o resultado apurado, registrando-se o esforço líquido empregado em cada atividade durante cada fase. É utilizado o fator de conversão do percentual de tempo produtivo sobre horas brutas, próprio de cada empresa, para fazer a transformação de horas líquidas em horas brutas, entrada para o cálculo da produtividade.

iii. Determinação da produtividade

- Para Desenvolvimento:

$$\text{Produtividade(PF/ mês-bruto/homem)} = \frac{\text{Produto do Trabalho(PF)}}{\text{Esforço do Trabalho(meses brutos)}}$$

O custo para desenvolver cada ponto por função do software é dado por:

$$\text{Custo(horas-brutas/homem)} = \frac{\text{Esforço do Trabalho(horas brutas)}}{\text{Produto do Trabalho(PF)}}$$

- Para manutenção/suporte:

$$\text{Produtividade(PF/ hora-bruta/homem no período de um ano)} = \frac{\text{Produto do Trabalho (PF) / Esforço do Trabalho (hs brutas)}}{\text{Produto do Trabalho (PF)}}$$

O custo para manter cada ponto por função do software no período de um ano é dado por:

$$\text{Custo(horas-brutas/homem)} = \frac{\text{Esforço do Trabalho (hs brutas)}}{\text{Produto do Trabalho (PF)}}$$

Para explicar o resultado de uma baixa ou alta produtividade, pode ser feita uma análise para determinar a possível influência dos seguintes fatores na produtividade: linguagem de codificação, tamanho do software, experiência da equipe, métodos estruturados, ambiente e ferramentas, qualidade da expansão (para desenvolvimento novo = 0,99), reutilização de código, circunstâncias geográficas, métodos de remoção de erros, recursos de codificação, organização da equipe e satisfação da equipe.

O método de Albrecht, ajustado às características de cada organização, é um excelente meio de se medir a produtividade do desenvolvimento. Porém, não é muito usado em sistemas de alta complexidade algorítmica e baixa complexidade de dados, porque, segundo C.Jones [8], é substancialmente incompleto quanto aos aspectos estruturais do software e ambíguo quanto às regras de contagem e ajuste.

Verner [35] aponta como a parte mais fraca do método a subjetividade de seus fatores de ajuste, os quais julga mais como atributos de custo que de tamanho. Desta forma, pontos por função se torna mais uma medida do esforço total que uma medida de tamanho, embora isto possa ser facilmente alterado, conforme observa.

Verner [35] critica, ainda, a deficiência dos fatores de ajuste usados para modificar os pontos por função totais, pelo fato destes só adicionarem pequenas correções com pesos máximos iguais, situação que é improvável no mundo real. Além disto, o efeito total é relativamente pequeno, de $\pm 35\%$, sendo o ajuste geral raramente maior que 15%.

Para dar maior precisão ao ajuste do método, Guerra [39] propôs uma extensão ao modelo de Albrecht, que chamou de modelo MACOS, incluindo 32 fatores de ajuste, representando aspectos técnicos de projeto e implementação, e aspectos associados ao ambiente de desenvolvimento. A seguir são apresentadas de forma sucinta as principais características deste modelo proposto por Guerra.

O modelo MACOS procura integrar, por meios de diferentes fatores de tamanho, os pontos de vista do usuário e do desenvolvedor, conforme a percepção deles. As fases do ciclo de vida cobertas pela estimativa feita através do modelo são: levantamento, análise de requisitos, projeto, codificação e teste, implantação e documentação final.

Como medida do esforço foi escolhido homens-hora para evitar diferenças do número de horas trabalhadas entre os meses do ano. Para cada homem-hora é atribuído um custo médio, e a partir daí o custo do projeto é estimado.

O principal fator considerado no custo é o tamanho do produto, medido em Pontos por Função Básicos (PFB). PFB corresponde à soma dos tipos medidos de: entradas do usuário, saídas para o usuário, consultas, arquivos lógicos internos e interfaces com outras aplicações.

A equação principal do modelo dá o esforço nominal (EN) em homens-hora como função dos PFB, e tem a seguinte forma:

$$EN = 7,41 * PFB^{1,19}$$

O esforço nominal é ajustado por um fator definido por:

$$EA = EN * FA,$$

onde: $FA = 1 + SGI/400$, e,

SDI = soma dos graus de influência dos fatores de ajuste nos custos.

A estimativa final de custo é o custo médio do homem-hora multiplicado pelo esforço ajustado.

Na validação das estimativas o erro médio entre as estimativas de esforço e os valores reais a partir dos quais o modelo foi desenvolvido foi de 25,8% e o desvio padrão foi de 17%. Foi feita também uma validação empírica com mais 20 projetos de diferentes tamanhos, com o objetivo de avaliar a validade predictiva do modelo, analisar o comportamento da métrica de tamanho utilizada e analisar o comportamento do fator de ajuste. Os dados dos projetos foram coletados seguindo uma cuidadosa metodologia e foram verificados quanto a qualidade do ajuste através da amplitude do erro relativo (teste MRE), conforme sugerido em Conte [31] e Kemerer [44].

Os resultados obtidos com as comparações entre os valores estimados e os reais mostraram que o modelo é apropriado para os projetos testados, embora tenha sub-estimado o esforço para os projetos maiores. Esta sub-estimação se deve a dois fatores:

- a métrica pontos por função não refletir adequadamente a complexidade interna do produto, e;
- a freqüente e pesada influência de fatores externos ao sistema, que geram perdas na produtividade da equipe.

II.3.2.3) MODELO DE PUTNAM (Rocha [2]) (Pressman [3])

(Pressman [64]) (Conte [31]) (Menezes [12])

É um método algorítmico composto desenvolvido por Putnam e Fitzsimmons ao final da década de 70. Putnam partiu da suposição

que a distribuição do esforço de pessoal no tempo correspondia a curva de Rayleigh, representada pela equação abaixo:

$$Y = k / t_d^2 \cdot t \cdot e^{-t^2 / 2 t_d^2}$$

onde Y = esforço em homens-ano no tempo t;

K = área em baixo da curva que representa o esforço total, em homens-ano, empregado no sistema durante todo seu ciclo de vida;

t_d = tempo necessário ao desenvolvimento (tempo em que o esforço é máximo).

O esforço de desenvolvimento, ED, é estimado como sendo:

$$ED = 0,4K, \text{ onde } k \text{ é o termo definido acima.}$$

A equação que descreve a relação entre tamanho do sistema, esforço de desenvolvimento, tempo de desenvolvimento e estado da tecnologia aplicada, foi derivada à partir da análise de projetos passados e tem a seguinte forma:

$$S_s = C_k^{1/3} \cdot K \cdot t_d^{4/3}$$

onde: S_s = número de instruções fonte entregues;

K = esforço no ciclo de vida em homens-ano;

t_d = tempo de desenvolvimento em anos;

C_k = constante para o estado da tecnologia aplicada.

A constante C_k reflete o efeito na produtividade de inúmeros fatores tais como restrições de hardware, complexidade do sistema, nível de experiência do pessoal e o ambiente de desenvolvimento. Putnam propôs um conjunto de vinte valores

variando de 610 a 57314 (assumindo K medido em programadores-ano e T em anos) dependendo da avaliação do nível do fator tecnologia aplicado ao projeto em questão.

Para calibrar a equação acima para um projeto passado com S_s , K e t_d conhecidos, Putnam sugere determinar C_k através desta equação, procurar na tabela o valor de C_k mais próximo do resultado encontrado e aplicá-lo à equação para determinar o valor estimado de K. Uma vez que o valor de C_k real é próximo do valor da tabela aplicado, é esperado que o valor real de K seja próximo do valor estimado para esta variável.

Para estimar o número de instruções do sistema, Putnam sugere que se faça três estimativas, a primeira na fase de definição e as outras duas durante o projeto.

- Primeira estimativa:

Deve ser feita no final da fase de definição do sistema, baseada em experiências anteriores e num conhecimento superficial do sistema. Seu objetivo é permitir avaliar a viabilidade do projeto, a partir do custo e tempo estimados. Estima-se:

a = número mínimo de instruções fonte

b = número máximo de instruções fonte

A partir daí determina-se o número esperado de instruções fonte,

$$F = (a + b) / 2,$$

o desvio padrão,

$$dF = (a - b) / 6,$$

e o tamanho esperado,

$$TE = F \pm dF.$$

A chance do tamanho real estar dentro de um desvio padrão da

média é de 68% e de estar dentro de três desvios é de 99% (distribuição probabilística).

- Segunda estimativa:

No início da fase de projeto, conhecidos os principais subsistemas, pode-se estimar o tamanho de cada um, que somados fornecerão o tamanho total do sistema. Isto é feito pelos membros da equipe que atuaram na definição, utilizando a técnica de Delphi. Para cada subsistema é estimado:

a = menor tamanho possível

b = maior tamanho possível

m = tamanho mais provável

O valor estimado de cada subsistema, F_i , é dado por:

$$F_i = (a_i + 4m_i + b_i) / 6$$

e o desvio padrão, dF_i , por:

$$dF_i = (a_i - b_i) / 6.$$

O tamanho total do sistema, F , é:

$$F = F_1 + \dots + F_n$$

e o desvio padrão, dF :

$$dF = ((dF_1)^2 + \dots + (dF_n)^2)^{1/2}.$$

- Terceira estimativa:

Na metade da fase de projeto, quando se pode dividir o sistema num número maior de subsistemas, aplica-se os mesmos procedimentos e obtem-se um valor mais aproximado.

Este modelo possui algumas relações que diferem fortemente dos outros modelos. Um exemplo disto é a relação que permite avaliar o esforço total versus o tempo de desenvolvimento. Colocando-se K em evidência na equação de cálculo do tamanho, pode-se observar que, para um dado tamanho e ambiente de

software, o esforço K varia inversamente com a quarta potência do tempo de desenvolvimento.

Esta penalidade tão severa imposta reduzindo-se o tempo de desenvolvimento não é largamente endossada por outros pesquisadores. Boehm, por exemplo, estima que reduzindo o tempo em 20 a 25%, o esforço total pode aumentar em 23%. Ele alerta, também, que uma compressão de mais de 25% não é viável. Estudos do próprio Putnam sobre 750 projetos não deram suporte à relação.

Outro exemplo é a equação de tempo mínimo em função do esforço, que pode ser derivada da teoria. A teoria de Putnam é baseada na equação de tamanho (S_s), na relação entre produtividade (L) e dificuldade (D), $L = c_1 D^{-2/3}$ onde c_1 é uma constante qualquer, e na restrição $(K / t_d^3) = \alpha$. A teoria afirma que a constante " α " toma valores discretos correspondentes a dificuldade do sistema a ser desenvolvido, incorporando o modelo seis valores que variam de 7,3 à 89,0.

A partir desta restrição, para K e t_d em meses e dado que $E=0,4K$, temos que:

$$\begin{aligned} t_d &= 12 \cdot ((k/12) / \alpha)^{1/3} = 12 \cdot ((E/(0,4 \cdot 12)) / \alpha)^{1/3} = \\ &= (12 / (4,8\alpha))^{1/3} \cdot E^{1/3} = c \cdot E^{1/3} \end{aligned}$$

A equação acima dá o tempo mínimo requerido para completar um projeto como função do esforço e se assemelha na forma as equações propostas por Walston e Félix e por Boehm. Para $\alpha = 27$, por exemplo, o tempo de desenvolvimento se torna:

$$t_d = 2,4E^{1/3} \quad \text{substituindo na equação acima}$$

$$t_d = 2,5E^{0,35} \quad \text{para Walston-Felix, obtido por mínimo}$$

quadrado

$t_d = 2,5E^{0,38}$ para Boehm, considerando aplicativo.

Esta restrição implica que o tempo de desenvolvimento não pode ser reduzido mais que o dado pela equação sob pena de prejudicar todo o projeto.

O modelo de Putnam também produz uma solução para tempo mínimo através da resolução simultânea das equações de tamanho e da restrição acima, assumindo poderem ser selecionados C_k e " α " a partir das características do projeto e existindo disponível uma estimativa de tamanho S_s :

$$K^{1/3} t_{dmin}^{4/3} = S_s / C_k$$

$$K / t_{dmin}^3 = \alpha$$

Conte [31], após as investigações acima descritas sobre o modelo de Putnam, aponta as seguintes limitações:

- O modelo trabalha razoavelmente bem para sistemas muito grandes, mas superestima o esforço para sistemas médios e pequenos;

- O efeito da compressão no tempo de desenvolvimento parece ser exagerada;

- O modelo oferece, além dos atributos tamanho e tempo de desenvolvimento, apenas a constante C_k , que deve refletir todos os outros atributos, como complexidade, uso de práticas modernas de programação, capacidade do pessoal, etc;

- Para usar este método o usuário deve fornecer um nível de Fator de Tecnologia (FT de 0 à 22) do qual será obtida a constante C_k . Porém, o esforço estimado é muito sensível a escolha de FT e uma diferença de 1 na sua escolha pode implicar

numa diferença de 100% ou mais no esforço estimado;

- Validações independentes não confirmaram a relação entre produtividade e dificuldade;

- A rigidez da relação esforço e tempo da curva de Rayleigh não pode ser aplicada geralmente no início do projeto e na fase de manutenção e não reflete variações entre as fases, que podem ocorrer quando se altera o grau de influência relativo de alguns fatores. Por exemplo, querer reduzir o tempo de desenvolvimento aumentando os recursos de pessoal ou equipamentos só funciona até aproximadamente 75% do tempo decorrido, segundo Brooks [46].

II.3.3) CONSIDERAÇÕES SOBRE OS MODELOS DE ESTIMATIVA DE CUSTOS

As entradas requeridas e as saídas produzidas por cada modelo variam muito. Além disto, cada um considera um conjunto diferente de características do software como parâmetros e ignora ou faz suposições sobre outras. Nem todos são aplicáveis a todas as classes de software e a todos os ambientes de desenvolvimento. É possível, mesmo, que nenhum capture as particularidades do ambiente de desenvolvimento de software de uma determinada organização. Neste caso, pode ser necessário combinar mais que um modelo (Navlakha [36]).

Para avaliar a aplicação destes modelos de estimativa de custo a diferentes ambientes de desenvolvimento muitos estudos têm sido conduzidos, entre os quais destacamos Navlakha [36], Kusters [38], Conte [31] e Abdel-Hamid [60], sobre os quais faremos breve descrição a seguir.

O trabalho de Navlakha [36] teve como objetivo determinar qual modelo ou combinação de ³modelos é mais apropriada para a

estimativa de custos num ambiente particular. Para isto foi conduzido um experimento que utilizou dados de sete projetos comercialmente desenvolvidos, da categoria de sistemas operacionais e ferramentas. Esses projetos eram pertencentes a duas organizações diferentes, que utilizavam o modelo COCOMO intermediário para utilitários para estimar o custo de seus projetos. Sete modelos de estimativa de custos foram escolhidos para o teste: as três formas do modelo COCOMO intermediário de Boehm, o modelo SLIM de Putnam, o modelo SEER de Jensen, o modelo Doty e o modelo Wolverton da IBM.

Através de experimentos estatísticos, a estimativa de cada um dos modelos para cada projeto e os resultados reais foram analisados. Estes experimentos consistiram de:

- Correlação entre os valores dos esforços estimados e real, que mostrou que a correlação entre o esforço real e o esforço estimado foram mais altas para o modelo SLIM e para o modelo de Jensen, bem como demonstrou alta correlação entre estes dois modelos, entre o modelo Doty e o modelo da IBM e entre os três modelos COCOMO, que tiveram comportamento distinto dos demais e correlação mais baixa com o esforço real.
- Como cada modelo enfoca aspectos diferentes do processo de desenvolvimento do software, foram feitas análises de regressão com o esforço real como variável dependente e os esforços estimados como variáveis independentes. A regressão só requereu uma etapa e a variável que iniciou a regressão foi o esforço SLIM, indicando assim, que este modelo é o mais apropriado para as duas organizações envolvidas na pesquisa. Foi observado também que o esforço estimado pelo modelo de Jensen também se aproxima bastante do real;

- teste de normalidade de Shapiro-Wilk nos resíduos (esforço real - esforço estimado pela regressão) para verificar se os dados residuais da estimativa de custos são normalmente distribuídos. Não foram encontradas evidências que indicassem que os resíduos não são normalmente distribuídos;
- para testar a metodologia em relação a sua capacidade de estimação foi retirado o maior dos projetos e feita nova regressão do esforço real contra os esforços estimados. A equação de regressão derivada foi aplicada para prever o esforço para o projeto retirado, obtendo-se um resultado impressionantemente positivo.

Como conclusão, Navlakha recomenda que antes de escolher um modelo de estimativa de custos, o gerente do projeto deve executar alguns experimentos estatísticos para verificar a validade de usar aquele modelo em seu ambiente de desenvolvimento. O gerente deve examinar, também, as suposições inerentes de cada modelo individual e balanceá-las contra seu ambiente de desenvolvimento para justificar sua seleção.

O trabalho de Kusters [38] também é experimental e consiste de testes e avaliações de versões automatizadas de modelos de estimativa de custos. Para isto foram definidos como critérios preliminares de seleção de modelos para a experiência as seguintes características:

- disponibilidade de versão automatizada, atualizada e com suporte comercial;
- o modelo deve ser baseado em projetos de sistemas de informação;

- o modelo não deve usar a unidade "LOC" (Lines Of Code) como entrada, devendo ser aplicável a estágios iniciais do desenvolvimento de um sistema de informação.

Baseado nesses critérios, foram selecionados quatro modelos: Before You Leap (BYL), ESTIMACS, SPQR20 e BIS/Estimator. A etapa seguinte consistiu do desenvolvimento de uma lista ponderada de vinte requisitos, elaborada com a cooperação de possíveis futuros usuários dos modelos. A partir desta lista foram eliminados os modelos SPQR20, por não ser calibrável, e o modelo BIS/Estimator, por não ser aplicável a estágios iniciais.

A segunda parte do estudo envolveu catorze experientes líderes de projetos, com os seguintes objetivos:

- determinar a precisão do modelo numa situação semi-real;
- determinar a aceitabilidade do modelo;
- determinar se a unidade "LOC" pode ser usada em estágios iniciais de desenvolvimento como bom indicador do tamanho do produto a ser desenvolvido.

Foram feitas estimativas para um projeto pelos catorze líderes de projeto e através dos dois modelos selecionados, e comparadas as estimativas obtidas com o resultado real. As conclusões foram baseadas nos resultados quantitativos e nas opiniões dos líderes de projeto e demonstraram que nenhum dos dois modelos pode ser usado em estágios iniciais de desenvolvimento. Apesar disto, os modelos foram sentidos como úteis pelos envolvidos na experiência, principalmente como "checklist" e meio de comunicação, e foi observado, ainda, que a calibração é essencial.

O trabalho de Conte [31] consiste da avaliação crítica de

vários modelos de estimativa através de critérios desenvolvidos por ele. Os critérios se dividem em critérios subjetivos (validade, objetividade, facilidade de uso, robustez e transportabilidade) e critérios objetivos ($MRE \leq 0,25$ e $PRED(0,25) \geq 0,75$, onde MRE é a amplitude média do erro relativo e $PRED(1)$ é a porcentagem de amostras com $MRE = 1$).

Os modelos avaliados foram agrupados em quatro categorias compreendendo: modelos históricos-experimentais, modelos baseados em estatísticas, modelos baseados em teoria e modelos compostos.

Na categoria modelos históricos-experimentais foi destacado o modelo Wolverton da TRW. Na categoria modelos baseados em estatísticas foram incluídos o modelo SDC, o modelo de Fan e Zagorski, o modelo de Walston-Felix, o modelo de Bailey-Basili, o modelo COCOMO, em seus três modos, e o modelo Doty. Como modelos baseados em teoria foram avaliados o modelo de Putnam, o modelo de Jensen e o modelo SSEM. Na categoria modelos compostos foram incluídos o modelo de Putnam, o modelo RCA PRICE-S, o modelo COCOMO, o modelo Softcost e o modelo COPMO, proposto pelo autor.

O modelo COCOMO foi julgado como o mais completo, o mais prontamente disponível e o mais fácil de usar. Outros modelos são apontados como compreensíveis no escopo e prontamente acessíveis: o modelo SLIM de Putnam e o modelo RCA PRICE-S, ambos produtos comercializados.

Foram encontradas dificuldades para comparar a performance de modelos em bancos de dados independentes devida a não disponibilidade das informações requeridas, sendo observado que o impedimento primário para uma boa pesquisa nesta área é a falta

de bancos de dados de projetos de desenvolvimento que tenham qualidade.

As comparações de performance dos modelos obtidas através das medidas objetivas de precisão definidas demonstraram que na maioria das vezes nenhum dos modelos avaliados foi capaz de atingir, em bancos de dados independentes, o nível de precisão definido, apesar de alguns serem capazes de atingir boa precisão em seus próprios bancos de dados de projetos.

Conte conclui que um modelo composto compreensível e preciso pode ser obtido combinando as melhores ferramentas de vários modelos, a exemplo de outras áreas científicas, e propõe o modelo COPMO nessa linha.

O trabalho de Abdel-Hamid [60] estudou o impacto da estimativa de tempo no comportamento do projeto, analisando a política de uso de fator de segurança.

Para isto foi escolhida a simulação experimental em cima do Modelo de Sistema Dinâmico de Gerência de Projetos de Desenvolvimento de Software (Abdel-Hamid [59]). A simulação foi escolhida devido aos altos custos e ao tempo necessário para conduzir experiências práticas.

O modelo permitiu simular o impacto da estimativa de tempo no comportamento do projeto e o cenário escolhido foi um ambiente de desenvolvimento real, no qual os gerentes de projeto eram recompensados tendo por base a precisão de suas estimativas. As estimativas eram feitas através do COCOMO, sendo que um fator de segurança era adicionado ao esforço estimado para cálculo da duração estimada do projeto.

A simulação revelou que o uso de fatores de segurança mais altos geravam alocações de esforços maiores; que não levavam a melhores resultados devido a perdas de produtividade; e ao maior consumo de recursos.

As principais conclusões tiradas da simulação são que:

- 1) Estimativas diferentes criam projetos diferentes
- 2) Estimativas mais precisas não são necessariamente estimativas melhores.

Os resultados dos estudos acima descritos e de outros estudos empíricos como o de Kemerer [44] e o de Mohanty [45] não conseguiram demonstrar que os modelos de estimativa de custos existentes são precisos e podem ser usados para prever esforço e tempo de desenvolvimento de projetos de software.

Desta forma, podemos concluir que o estado da arte corrente dos modelos de estimativa de custo não nos permite confiar plenamente em seus resultados. De qualquer forma, estes modelos têm seu papel numa organização como um todo e podem ser usados para complementar o julgamento de especialistas, conforme observa Ghezzi [17].

Muitas propostas têm surgido para suprir deficiências em determinados aspectos dos modelos de estimativa de custos. Um aspecto que tem sido alvo de vários estudos é a relação entre tempo versus esforço de desenvolvimento e seu impacto na produtividade e nos custos. Outro aspecto deficiente que tem sido estudado é o fato da maioria dos modelos existentes para estimar recursos basear a estimativa de esforço no tamanho do código.

Tamanho de código, que não é uma métrica ideal para produtividade, parece ser apropriada para determinar o custo total do ciclo de vida do software, tanto no início como nos estágios mais avançados do projeto, segundo Ghezzi [17]. O problema é que, para estimar o tamanho não dispomos de métodos que alcancem pelo menos a confiabilidade dos modelos existentes para estimar custos. Assim, a estimativa de tamanho é o primeiro e mais fraco elo na cadeia da estimativa de custos (Verner [35]).

Para solucionar essas deficiências dos modelos de estimativa de recursos, muitos estudos têm sido conduzidos e algumas propostas promissoras têm surgido, entre as quais destacamos:

- o trabalho de Ho [37], que propõe um processo que permita ao gerente de projetos avaliar diferentes combinações de quantidades de esforço e tempo de desenvolvimento que maximizará a produtividade ou minimizará o custo para um dado software, e,
- o trabalho de Verner [35], que propõe um modelo de dimensionamento de software adequado para servir de entrada para modelos de estimativa de custos baseados em tamanho de código.

A seguir esses trabalhos são apresentados resumidamente.

Ho partiu de uma das mais importantes propriedades dos modelos de Putnam e de Boehm: "o produto de software, em termos de instruções fonte ou linhas de código, é diretamente dependente de dois fatores de entrada, o esforço em pessoas-meses e o tempo de desenvolvimento". Esta propriedade implica que diferentes combinações de esforço e tempo podem produzir um determinado nível de saída, ou que um dos fatores de entrada pode ser substituído pelo outro para manter constante o nível de saída.

No modelo de Putnam esta propriedade é claramente expressa em sua equação, que pode ser usada para analisar a relação esforço versus tempo de desenvolvimento, e seu impacto:

$$S_s = C_k K^{1/3} t_d^{4/3},$$

onde

S_s = número de linhas de código entregues do produto final;

C_k = constante representando o estado da tecnologia;

K = esforço no ciclo de vida, em pessoas-anos;

t_d = tempo de desenvolvimento, em anos.

Já através das equações do modelo COCOMO de Boehm não se tem uma medida direta do impacto do tempo de desenvolvimento no número desejado de instruções fontes. Porém, combinando suas equações, Ho demonstra que se pode obter uma equação que pode ser usada para analisar a relação esforço versus tempo, semelhante a de Putnam. A equação é obtida multiplicando-se as equações do COCOMO Básico e resolvendo-se para KDSI, derivando-se:

$$KDSI = 0,1815 (MM)^{0,5909} (TDEV)^{0,9523}.$$

Explorando a lei de Brooks [46], que diz que força de trabalho e tempo não são intercambiáveis devido a conseqüente queda de produtividade, em termos de taxas marginais de substituição de esforço por tempo, Ho observa que: "Mesmo que esforço e tempo não sejam intercambiáveis, ainda necessita-se conhecer como a saída (produto) pode ser maximizada para um dado custo, ou como o custo pode ser minimizado para um dado nível de saída, para obter a melhor taxa de esforço x tempo".

Assim, conclui que o custo do tempo de desenvolvimento não pode ser ignorado, e deve ser tratado durante a determinação do custo total de desenvolvimento do software.

Para determinar a combinação economicamente mais eficiente de tempo e esforço de desenvolvimento, Ho propõe uma equação para custo total de desenvolvimento, que inclui tanto um valor para o custo do esforço de desenvolvimento em pessoas-meses como para tempo de desenvolvimento. Esta equação pode ser expressa como:

$$\text{Custo} = P_m(\text{MM}) + P_t(\text{TDEV}),$$

onde

P_m = custo de homens-meses ou "man-months" (MM)

P_t = custo do tempo de desenvolvimento (TDEV)

Assim, Ho demonstrou que, tendo-se o custo unitário do esforço e do tempo e aplicando a equação derivada do modelo COCOMO ou a equação do modelo SLIM, é possível determinar que combinação de esforço e tempo de desenvolvimento é mais economicamente eficiente para produzir um produto de determinado tamanho.

Para determinar o custo do tempo, Ho propõe que os seguintes fatores devam ser considerados como determinantes significativos deste custo:

1. Custo de oportunidade entre comprar e desenvolver;
2. Custo da estrutura (organização) que deve incluir o custo de preparo da linha de montagem e os custos de testes e problemas esperados nos testes. Estes custos vão incorrer independente do sistema ser desenvolvido internamente ou não;
3. Custos gerados por atrasos, como tentativas especiais para manter o cronograma ou perdas na renda. O custo da oportunidade deve ser medido como o valor presente das perdas esperadas resultantes de um atraso. Um exemplo é o caso de multas fixadas

em contratos como penalidade por atrasos.

4. Custos de alocar, realocar e treinar durante o desenvolvimento devido a atritos de pessoal ou a restrições absurdas de prazo, que levam a perda de membros valiosos da equipe.

5. Custos de retenção, como: seguro, deterioração, interesse e oportunidade de investir em atividades de desenvolvimento. Quanto maior o tempo de desenvolvimento maior os custos de retenção.

No trabalho de Verner [35] é analisado o fato da estimativa de tamanho e de custos serem parte de um processo iterativo de tomada de decisão envolvendo questões entre especificação do sistema, aspectos de qualidade do software e preparação de cronogramas com restrições de recursos, apresentando, porém, a primeira deficiências em relação a segunda, como pode ser visto na figura abaixo.

Fig.II.3: Estimativa de Tamanho e Custos de Software.

Fonte: Verner [35]

Analisando algumas tentativas de resolver o problema da estimativa de tamanho, como a Análise dos Pontos por Função (Albrech [32]) e o Sistema Bang (DeMarco [5], [42]), Verner aponta deficiências em ambos. Na Análise dos Pontos por Função seus fatores de ajuste são considerados como a parte mais fraca

do método. No Sistema Bang o problema apontado reside na falta de informações suficientes para suprir o baixo nível de suas primitivas funcionais, aliado ao fato do trabalho necessário para uma organização construir seus próprios pesos e classes de primitivas, conforme é sugerido por DeMarco, inibir seu uso na presente forma.

Verner considera, porém, que uma abordagem similar ao sistema Bang pode ser promissora se for possível construí-la como produto de uma ferramenta automatizada de especificação baseada na análise estruturada de sistemas.

Após concluir que os três principais problemas da estimativa de tamanho são a função do sistema, a dependência da linguagem de programação e as restrições ou critérios do projeto, e julgando que o isolamento destes problemas parecem ser essenciais para entender o processo de dimensionamento, Verner propôs um modelo de dimensionamento de software adaptável para servir de entrada para modelos de estimativa de custos baseado em tamanho.

A arquitetura deste modelo (Fig. II.4 a seguir) torna explícito no cálculo do tamanho em KDSI de um sistema: o papel do modelo abstrato do sistema (tamanho A), da linguagem ou linguagens empregadas (tamanho A -> tamanho B), e de outros atributos de tamanho (tamanho B -> tamanho C).

Fig.II.4: Um Modelo de Estimativa de Tamanho e Custos de SW.

Fonte: Verner [35]

As entradas para o modelo são o modelo do sistema, que deve ser uma especificação abstrata livre de restrições de implementação e outras, e as restrições do sistema, que são os requisitos não pertencentes a categoria anterior que podem ser considerados como atributos de tamanho, como por exemplo: interatividade, amigável ao uso, clareza dos programas, descentralização ou distribuição do sistema.

O principal objetivo da parte A do modelo é a objetividade. Assim, é proposto que o tamanho A seja calculado diretamente como produto de um sistema automatizado de definição do processo, e que sua unidade de medida seja uma das seguintes unidades: LOC de uma linguagem padrão, ou uma medida independente da linguagem como Pontos por Função e Sistema Bang, ou ainda, uma métrica similar melhorada que supere as deficiências destes dois métodos.

A parte B do modelo de dimensionamento de Verner mapeia o tamanho A, independente de linguagem, para um tamanho dependente da linguagem, chamado tamanho B, que pode ser entendido como o

tamanho do código neutro de um sistema para uma linguagem escolhida. Quando usadas várias linguagens ou quando o mapeamento é diferente para partes claramente definíveis do sistema, tamanho B será a soma ponderada dos componentes. Para isto, as noções de níveis de linguagens e taxas de expansão, trabalho similar ao de Albrech [32] relacionando Pontos por Função e LOC para diferentes linguagens, é levado em consideração.

A parte C do modelo aplica fatores de correção apropriados ao tamanho B, ou parte dele, para obter um resultado realístico em KDSI. São consideradas nesse ajuste restrições ou decisões do projeto e a condição sugerida no trabalho de Weinberg and Schulman, descrita por Verner como sendo premissa que os fatores de ajuste sejam maiores que o limite de Pontos por Função de $\pm 35\%$, observado o fato que, na maioria dos sistemas grandes, ajustes são aplicados apenas a partes do sistema, como a interface com o usuário, por exemplo.

O modelo de Verner de dimensionamento de software vem sendo refinado e dados vêm sendo coletados para determinar seus parâmetros e fazer sua validação em ambientes comerciais.

II.4) FERRAMENTAS PARA GERÊNCIA DE PROJETOS

O suporte automatizado para o processo de desenvolvimento de software torna-se imprescindível quando os sistemas atingem uma complexidade que torna necessário o envolvimento de vários profissionais empenhados na tarefa de desenvolvê-lo.

Para dar o suporte necessário torna-se imprescindível a existência de ferramentas automatizadas de apoio aos métodos. Ferramentas são recursos automatizados que permitem a realização das atividades envolvidas no processo de desenvolvimento de

software (Aguiar [23]).

A quantidade de trabalhos de produção de ambientes automatizados de gerência de projetos de software adaptáveis a diferentes métodos ainda é pequena. Hurst [19] relaciona alguns trabalhos nesta área, entre os quais:

- o projeto CONCERTO para especificar e construir um ambiente integrado e flexível de engenharia de software através de parametrizações de ferramentas;
- o projeto SAGA, que permite a especificação formal do ciclo de vida que é usado no ambiente e a descrição das práticas de trabalho, através de gramática gerencial;
- o ambiente SIGL (ISEF), que inclui a ferramenta GUIDE com "agendas" de trabalhos a serem feitos, trabalhos em progresso e trabalhos completados;
- o ambiente Apollo Domain, que inclui um gerente monitor das dependências definidas pelo usuário e um gerente aconselhador, que mantém as informações gerais do projeto e fornece modelos para refazer tarefas comuns.

Mais recentemente novos trabalhos têm surgido, como o projeto SPMMS ("Software Production and Maintenance Management Support") de Hurst [19], o ambiente ISTAR ("Integrated Project Support Environment") de Dixon [61] e a ferramenta de planejamento inteligente de Maddy [22].

Além destes ambientes, muitas ferramentas existem, também, que apoiam determinados aspectos da gerência de projetos. As principais são implementações e extensões de métodos de estimativa de custos e ferramentas de gerência da configuração do software. Lederer [68] identifica como as mais importantes:

Estimacs, Spectrum/Estimator, Inhouse package, Project Workbench, Nolan/Prompt, Project Manager, AGS PAC III, DEC/VAX Software Project Manager e Microsoft Project.

A seguir descreveremos o ambiente ISTAR e algumas ferramentas de estimativa de recursos.

II.4.1) ISTAR (INTEGRATED PROJECT SUPPORT ENVIROMENT)(DIXON [61])

O sistema ISTAR oferece facilidades para o planejamento, monitoração e controle de projetos de software, partindo do princípio que o suporte gerencial deve ser aplicado às seguintes áreas:

1) Administração do projeto, que envolve:

- O planejamento, estimativa e programação das atividades necessárias para atingir os níveis de qualidade, funcionalidade e performance desejados, respeitadas as restrições de recursos, custos e escalonamento;
- atribuição das atividades à equipe, e;
- aplicação de controles.

2) Gerência dos recursos, que envolve:

- a identificação dos recursos disponíveis;
- a alocação apropriada do pessoal, e;
- a resolução de conflitos sobre o uso de um determinado recurso.

3) Gerência dos custos, que envolve:

- a comparação das informações sobre o consumo de recursos previsto e real;
- a estimativa do progresso e desenvolvimento de controles.

A proposta geral do sistema ISTAR é mostrada na figura a seguir:

Fig.II.5: Gerência do Projeto, dos Custos e dos Recursos

Fonte: Dixon [61]

A principal característica da abordagem do sistema ISTAR é a decomposição de qualquer tarefa em sub-tarefas ou atividades, constituindo pacotes de trabalho bem definidos e de mais fácil entendimento, denominados contratos. Cada contrato é executado

por um desenvolvedor para um cliente, seja o cliente um gerente ou outro desenvolvedor.

Esses contratos são independentes, possuindo base de dados específicas criadas em sua definição. Os contratos podem ser decompostos gerando uma hierarquia de contratos. Cada subcontrato conserva a mesma independência e às suas tarefas podem ser alocadas ferramentas que atuarão apenas na base de dados específica deste subcontrato.

II.4.2) MÉTODOS PARA ESTIMATIVA DE CUSTOS APOIADOS POR COMPUTADOR

São ferramentas automatizadas para estimar custo e esforço, e executar análises simuladas para variáveis importantes do projeto. Suas características básicas são (Pressman [3] Pressman [64]):

- permitem estimativas quantitativas e funcionais do tamanho do software;

- permitem contemplar características qualitativas do projeto tais como complexidade, confiabilidade requerida ou criticalidade na empresa; e

- permitem descrever a equipe e o ambiente de desenvolvimento.

A partir desses dados a ferramenta pode gerar estimativas de esforço e custo, distribuição de pessoal, desenvolver cronogramas e fazer análise de riscos, em alguns casos.

Como exemplos deste tipo de ferramenta selecionamos o SLIM, o ESTIMACS e três ferramentas desenvolvidas na COPPE/UFRJ para microcomputador IBM PC: ESTIME, SAACS e SGPS.

a) SLIM (PRESSMAN [3])

Putnam incorporou seu modelo neste produto, que aplica, também, programação linear, simulação estatística e técnica de PERT, para gerar as estimativas do projeto.

Permite ao planejador executar as seguintes funções, numa sessão interativa:

- calibrar o modelo ao ambiente local de desenvolvimento, interpretando dados históricos fornecidos pelo planejador;
- criar um modelo de informação do software a ser desenvolvido, através da elucidação das características básicas do software, atributos de pessoal e considerações sobre o ambiente;
- dimensionar o software.

Após estabelecer o tamanho do software, SLIM calcula seu desvio padrão. O planejador pode invocar uma análise de programação linear que considera restrições no custo e esforço. SLIM fornece também uma distribuição de esforço e custo mensal, usando a curva de Rayleigh-Norden como modelo, permitindo que se possa planejar os recursos de pessoal e preparar um fluxo de caixa para o projeto.

b) ESTIMACS (PRESSMAN [3])

Modelo de macro estimação que usa o método Pontos por Função para acomodar uma série de fatores que influenciam nos custos.

Contém um conjunto de modelos que habilita o planejador estimar os aspectos abaixo:

1. Esforço de desenvolvimento

O modelo de esforço combina dados sobre o usuário, o

desenvolvedor, a geografia do projeto e número de funções principais a serem implementadas, com dados do domínio da informação requerida para o cálculo dos pontos por função. Leva em conta a natureza geral do sistema proposto, sua complexidade, desempenho e confiabilidade requeridas.

2. Custo e recursos de pessoal

Usando dados do modelo de esforço, ESTIMACS gera a distribuição de pessoal e custo usando um banco de dados customizável.

3. Configuração de hardware

A configuração do hardware requerido é dimensionada usando uma série de respostas a questões que ajudam o planejador avaliar dados como volume transacional e aplicações.

4. Risco

O nível de risco associado com o sucesso da implementação é determinado baseado nas respostas de um questionário que examina fatores como tamanho, estrutura e tecnologia aplicada.

5. Efeitos de projetos simultaneamente em desenvolvimento

ESTIMACS leva em conta também efeitos de projetos que estão sendo desenvolvidos concorrentemente.

c) ESTIME (MENEZES [12])

ESTIME é uma ferramenta para estimar custos de desenvolvimento de software proposta por Menezes [12], baseada no modelo COCOMO. Utiliza estimativa por analogia e julgamento de especialista para facilitar a estimativa do tamanho do projeto.

O método de Putnam é, também, utilizado para o cálculo do número de instruções fonte gerando estimativas com precisão

crescente, identificando as respectivas margens de erro em três etapas do desenvolvimento: na geração da Proposta de Desenvolvimento, após concluída a Especificação e iniciada a geração do Plano do Projeto, e ao final da subfase de Projeto da Arquitetura.

Usando o COCOMO, ESTIME permite que se tenha a visão a nível de sistema, dos métodos "top-down", e a nível de subsistemas, dos modelos "bottom-up", permite obter, também, a estimativa por ano de manutenção e, dependendo do modelo COCOMO escolhido, permite que o gerente realize análise de sensibilidade dos atributos de custo.

d) SAACS (RIBEIRO [11])

SAACS é um sistema especialista de apoio à decisão desenvolvido por Ribeiro [11], para apoiar a tarefa de estimativa de custos de desenvolvimento de software baseada no julgamento de especialistas. Para isto, o sistema é capaz de armazenar o conhecimento de especialistas da área de forma a poder auxiliar seus usuários na tomada de decisão.

A partir das respostas fornecidas pelo usuário a uma série de perguntas feitas pelo sistema a respeito do software cujo custo se quer estimar e do ambiente onde será desenvolvido, o sistema consulta sua base de conhecimentos, que possui regras de auxílio a estimativas de custo e orientações para reduzir o custo final de um projeto, e apresenta suas conclusões. O especialista, então, estima o custo baseado nas conclusões do sistema e na sua própria experiência.

O sistema é capaz de armazenar informações de uma sessão na sua base de dados histórica, para reutilização. Além disto,

possui um mecanismo de "what if" para permitir explorar o comportamento sob novas condições. Fornece, ainda, dois tipos de explicação ao usuário: explicação dos motivos de uma pergunta e suas regras relacionadas e explicação para o significado de cada resposta possível.

e) SISTEMA DE GERÊNCIA DA PRODUÇÃO DE SOFTWARE (MACHADO [13])

Machado [13] propôs um ambiente de gerência da produção de software com os seguintes objetivos:

- determinar o tamanho do software a ser desenvolvido pela metodologia pontos por função;
- calcular a produtividade de programas, módulos, projeto e, anualmente, da empresa, tanto para manutenção quanto para desenvolvimento em geral;
- determinar o esforço e o tempo total do projeto, das fases, das tarefas e das atividades;
- permitir controlar o andamento do projeto à luz dos planos elaborados, prazos previstos e marcos estabelecidos.

O sistema foi dividido nos seguintes processos:

1. Cálculo do tamanho, que envolve:
 - a) Identificação das propriedades do software
 - b) Cálculo do produto do trabalho
2. Cálculo da produtividade, que compreende:
 - a) Registro do esforço do trabalho diário
 - b) Encerramento do desenvolvimento
 - c) Apuração da produtividade para cada um dos níveis
3. Elaboração das estimativas do desenvolvimento, que compreende:
 - a) Cálculo do esforço total

- b) Cálculo do tempo total
 - c) Cálculo da demora das fases
 - d) Cálculo da demora das tarefas
 - e) Cálculo da demora das atividades
4. Acompanhamento do desenvolvimento (não implementado):
- a) Elaboração dos planos de desenvolvimento
 - b) Avaliação do andamento geral em relação aos planos

II.5) CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados conceitos sobre gerência de projetos, foram detalhadas as atividades principais da gerência de projetos, planejamento e acompanhamento, e discutidos métodos e ferramentas para a gerência de projetos de desenvolvimento de software.

A partir desta pesquisa observamos que não existem panacéias para solucionar os problemas da produção de software. A aplicação do processo certo, da tecnologia certa, da metodologia certa e das ferramentas certas, ajudam a controlar as dificuldades do desenvolvimento de software mas não as eliminam.

A adequada gerência dos projetos de desenvolvimento de software, feita por profissionais que tenham não só habilidades para planejar, organizar e controlar projetos, mas, também, habilidade para liderança técnica, se torna fundamental para lidar com estas dificuldades e conter os custos crescentes da produção de software, causados não pela demanda de produtos com crescente complexidade mas sim pela falta de capacitação na área gerencial.

Para dar capacitação gerencial, o sistema educacional na área de Engenharia de Software deve fornecer, também, educação

básica em gerenciamento de pessoas, técnicas gerenciais, sistemas motivacionais, relações humanas e técnicas de negociação, não se concentrando apenas nos aspectos técnicos de uma engenharia de software altamente sofisticada, muitas vezes não aplicável à maioria dos projetos.

Além disto, é necessário ainda um esforço maior de aproximação entre os centros de pesquisa e as organizações de software para que possam ser desenvolvidos modelos mais realísticos de gerência de projetos que possam melhorar a produtividade e eficiência do processo de produção de software.

Entendemos que um sistema completo para gerência de projetos de desenvolvimento de software deve conter, além de ferramentas de gerência, apoio para a tomada de decisão. Este apoio pode ser dado através de conhecimento de especialistas armazenado no sistema. Neste tipo de problema, porém, ainda não é possível deixar que todas as decisões sejam tomadas pelo sistema. A intervenção do usuário é essencial pelo fato do conhecimento não estar suficientemente claro. Assim, o sistema deve fornecer apoio à decisão, sem, contudo, pretender dar a palavra final sobre qualquer questão.

Este é o caso dos sistemas de apoio à decisão baseados em conhecimento, assim chamados porque incorporam características de sistemas baseados em conhecimento e de sistemas de apoio à decisão.

Sistemas de apoio à decisão, de acordo com Yourdon [66], são sistemas que não tomam decisões por eles mesmos, mas auxiliam seu usuário a tomar decisões inteligentes e bem informadas sobre vários aspectos de uma operação.

Sistemas baseados no conhecimento contêm conhecimentos para utilização em determinada tarefa. Empregamos o termo sistemas baseados no conhecimento em detrimento do termo sistemas especialistas, porque este último tem obtido conotações negativas, por tais sistemas não atenderem as expectativas geradas nos primórdios da inteligência artificial (Wielinga [75]).

Conforme Feigenbaum [67], os sistemas especialistas são uma espécie de sistemas baseados no conhecimento, embora os dois nomes sejam muitas vezes empregados indistintamente. O termo Sistemas Baseados no conhecimento surgiu como alternativa mais apropriada por se referir a um escopo maior de sistemas e, por constituir uma proposta menos pretensiosa sobre a capacidade de tais sistemas (Wielinga [75]).

Sistemas baseados no conhecimento, independente do modelo de arquitetura, possuem os seguintes componentes comuns (Luconi [73], Hart [74]):

a) base de conhecimento

É o repositório do domínio do conhecimento, que pode ser armazenado de diferentes formas, como por exemplo: regras de produção, redes semânticas e "frames".

b) mecanismo de inferência

Contém os métodos através dos quais o conhecimento é manipulado. O paradigma de inferência do motor é a estratégia de pesquisa utilizada para desenvolver o conhecimento necessário. Vários paradigmas são adotados num sistema baseado no conhecimento, como por exemplo: orientação a objetos, meta-controle e gerenciamento de incerteza. Grande parte, porém, está

baseada em dois conceitos fundamentais: "backward chaining", que é um processo de raciocínio "top-down", que inicia dos objetivos desejados e trabalha na direção das condições de requisito, e, "forward chaining", que é um processo de raciocínio "botton-up" que começa com condições conhecidas e trabalha em direção ao objetivo desejado.

c) Interface com o usuário

Permite o sistema se comunicar com o usuário.

No próximo capítulo apresentamos o Sistema de Apoio à Decisão Baseado em Conhecimento para Gerência de Projetos de Desenvolvimento de Software proposto para a Estação TABA. É apresentada, também, uma das ferramentas deste sistema, a Ferramenta para Planejamento de Recurso, que foi implementada como parte deste trabalho.

CAPÍTULO III

UM SISTEMA DE GERÊNCIA DE PROJETOS DE DESENVOLVIMENTO DE SOFTWARE PARA A ESTAÇÃO TABA

Este capítulo apresenta o sistema proposto com o objetivo de planejar e acompanhar projetos de desenvolvimento de software e a documentação da ferramenta desenvolvida para implementar uma de suas funções, o planejamento de recursos. O modelo de Plano de Projeto proposto neste trabalho, que serviu de base para definir as funções do sistema, encontra-se no Anexo I.

A Seção III.1 situa o sistema no contexto da Estação TABA com suas implicações. Na Seção III.2 apresentamos os requisitos do sistema. Na Seção III.3 apresentamos um WBS com a descrição das funções especificadas para o sistema. A Seção III.4 contém a documentação da ferramenta implementada. No Anexo I se encontra o modelo proposto de Plano do Projeto, o Anexo II contém o Dicionário de Dados complementando a especificação dos requisitos da ferramenta implementada e no Anexo III encontra-se o manual do usuário, para utilização da ferramenta.

III.1) O Sistema de gerência de Projetos no TABA

Como já mencionado no Capítulo I, o projeto TABA visa a construção de uma estação de trabalho para o desenvolvimento de software configurável para diferentes domínios de aplicação. Esta Estação tem como objetivos: auxiliar o engenheiro de software no planejamento e criação do ambiente mais adequado ao desenvolvimento de um produto específico, implementar as

ferramentas necessárias a este ambiente e permitir desenvolver e executar o produto (software) na estação (Rocha [9] Rocha [10]).

Estas funções determinam quatro ambientes na estação TABA: o meta-ambiente, o ambiente de desenvolvimento de ferramentas, o ambiente de desenvolvimento e o ambiente de execução (figura III.1)

Fig. III.1: Ambientes do TABA

Fonte: Rocha [30]

O meta-ambiente do TABA tem como principais objetivos planejar e gerar o ADS necessário ao desenvolvimento de um software particular. É um meta-ambiente do tipo configurador por ter a característica de gerar ambientes a partir da junção de componentes. Foram considerados como componentes a partir dos quais o ADS é formado: ciclo de vida, métodos, ferramentas e hardware. Dessa forma, planejar o ADS é determinar uma configuração adequada de ambiente para o desenvolvimento de um projeto.

Para gerar ADS's o meta-ambiente utiliza como recurso reunir componentes. Para isto, é composto por seis processos principais (Figura III.2), ilustrados a seguir:

Fig. III.2: Diagrama de fluxo de dados do meta-ambiente

Fonte: Aguiar [23]

O processo "Planejar ADS" já se encontra implementado através de XAMÃ. XAMÃ foi desenvolvido fundamentado em quatro questões, comentadas a seguir (Aguilar [23]):

1. Análise das características do projeto a ser desenvolvido

Para indicar os componentes que formarão um ADS é necessário identificar as características do projeto para o qual este ADS está sendo planejado. Uma base de dados de características de aplicações diversas apóia o usuário nesta tarefa.

2. Definição do ciclo de vida

Para escolher o ciclo de vida que possa ser adaptado a seu projeto o usuário conta com uma base de ciclos, representados por grafos que expressam suas particularidades e seu relacionamento com os demais componentes do ADS, e com uma base de conhecimentos, que contém informações que o auxiliam nesta escolha.

3. Suporte à decisão

O planejamento do ADS é um processo heurístico, necessitando de conhecimento de especialistas na área de aplicação e experiência no uso de métodos e ferramentas. XAMÃ contém conhecimentos para apoiar a indicação de componentes do ADS e três bases de dados para dar subsídios ao suporte à decisão: base de métodos e ferramentas, base de ciclos de vida e base de características de aplicações.

4. Reutilização de informações

Para aproveitar a experiência de outros desenvolvimentos, é gerado um histórico no final do projeto com todas as versões de ADS utilizadas e a razão que motivou cada descarte. A avaliação destes dados servirá para alimentar as bases de dados e de conhecimento.

As bases de dados e de conhecimento que dão suporte à decisão em XAMÃ foram compostas a partir das seguintes informações:

1. Base de dados de características de aplicações

As aplicações foram descritas a partir de atributos classificados como: características básicas, requisitos de qualidade e características de implementação.

As informações que constam na base de características de aplicações da versão de XAMÃ desenvolvida, foram obtidas a partir da análise de dados considerando as características básicas e os requisitos de qualidade.

2. Bases de ciclos de vida

A base de ciclos contém a descrição de diferentes ciclos de vida. A base de conhecimentos contém regras, que indicam ciclos possíveis de serem utilizados no projeto em questão, baseadas nos seguintes fatores que determinam o perfil do ciclo de vida: tipo de linguagem, tipo de suporte gerencial que se pretende adotar, recursos que possibilitem a reutilização, tipo da aplicação e o fato do ciclo ser para desenvolvimento ou para manutenção. A

versão de XAMÃ implementada considera as implicações relacionadas ao tipo de linguagem.

3. Bases de métodos e ferramentas

Constam de base de dados, com informações sobre características de métodos e ferramentas, e de uma base de conhecimentos, contendo regras para indicar métodos e ferramentas. A avaliação dos métodos considera a utilizabilidade de cada um para construção do produto, para avaliação do produto e para gerência do processo de desenvolvimento.

A versão de XAMÃ implementada consta de conhecimento para indicar métodos considerando os seguintes critérios de utilizabilidade para construção:

- aplicabilidade quanto ao tipo de processamento e nível de complexidade,
- expressividade quanto a abordagem de desenvolvimento utilizada e grau de formalidade que os instrumentos notacionais do método possibilitam.

No contexto do Projeto TABA, um ADS deve conter um conjunto integrado de métodos e ferramentas que suportem as atividades de construção do produto, as atividades de planejamento e avaliação da qualidade do produto (Rocha [62]) e as atividades de gerência do processo de desenvolvimento. Para isto, a Estação TABA contém três módulos básicos: o Módulo de Construção, o Módulo de Avaliação da Qualidade e o Módulo de Gerência.

Este trabalho tem como objetivo definir o Módulo de Gerência, que vem complementar e integrar XAMÃ a um ambiente

completo de gerência de projetos.

Para dar o suporte gerencial necessário a um projeto, o Sistema de Gerência será composto por ferramentas para planejamento e acompanhamento, integradas a bases de conhecimentos para apoiar a tomada de decisão. O sistema deve, ainda, gerenciar o fluxo de informações do projeto e auxiliar o gerente no planejamento de sua própria atividade de gerência, dando apoio para definir uma estratégia de gerência adequada para um projeto específico.

O sistema de gerência de projetos irá atuar em três ambientes da Estação, conforme a figura abaixo:

Fig. III.3: Ambientes do TABA onde atua o sistema proposto.

No meta-ambiente o sistema vai permitir analisar o ADS do ponto de vista gerencial e definir uma estratégia para gerenciar o projeto. A definição da estratégia para gerenciar o projeto implica em escolher os métodos e ferramentas a serem utilizados nas atividades de planejamento e acompanhamento do projeto, bem como as políticas, regras e procedimentos a serem aplicadas. Como produto é gerado o Plano de Gerência do Projeto, contendo estas definições organizadas em forma de documento.

No ambiente de desenvolvimento o sistema deve oferecer as ferramentas especificadas para planejamento e acompanhamento do projeto, de acordo com a estratégia definida no meta-ambiente.

As ferramentas de planejamento devem abranger:

- o planejamento detalhado do processo de desenvolvimento especificado em XAMã, adicionando as atividades gerenciais e marcos de controle;
- a estimativa de recursos de pessoal (hardware e software são especificados como parte do ADS através de XAMã);
- a escolha da estrutura organizacional adequada ao projeto e a alocação de pessoal às posições definidas nesta estrutura;
- a gerência dos riscos; e,
- a determinação da natureza do suporte que o projeto necessitará em termos de documentação, controle da qualidade e gerência da configuração.

Como produto destas atividades são gerados os demais planos componentes do Plano do Projeto, que está especificado no modelo proposto apresentado no anexo I.

Para dar suporte ao acompanhamento, o sistema deve:

- permitir a comparação contínua da realidade contra o planejado,

medindo o grau com que o progresso das tarefas, a geração das saídas e o consumo de recursos estão atendendo aos planos;

- apoiar a tomada de decisão, no caso de desvios do projeto em relação aos planos, fornecendo meios para avaliar as opções e suas prováveis consequências;
- dar indicações de condições correntes e prováveis de ocorrer no futuro, que podem afetar a qualidade do produto em relação a seus requisitos.

No ambiente de operação é feito o acompanhamento do produto desenvolvido. Para a gerência do produto o sistema deve oferecer facilidades para a gerência:

- de versões e,
- de propostas de modificação do produto, permitindo a avaliação do impacto, alocação de pessoal para executar as modificações e apoio aos demais aspectos envolvidos no processo de alterar um software.

A seguir apresentamos os requisitos e a especificação do Sistema de Gerência de Projetos de Desenvolvimento de Software proposto para a Estação TABA.

III.2) REQUISITOS DO SISTEMA

Foram definidos os seguintes requisitos para o Sistema de Gerência do TABA:

- 1) O sistema deverá ser configurável, como o é toda Estação TABA.
- 2) O sistema deverá ser integrado a outras ferramentas da estação (XAMã e Ambiente de Controle da Qualidade).

3) O sistema deverá ser evolutível, possibilitando integrar novas ferramentas (para gerência da configuração, de apoio a documentação, de apoio a análise de requisitos, etc).

III.3) ESPECIFICAÇÃO DO SISTEMA PROPOSTO

Acompanhando o modelo de Plano de Projeto proposto, que se encontra no Anexo I deste trabalho, o Sistema de Gerência de Projetos de Desenvolvimento de Software deve conter as seguintes funções descritas a seguir:

Fig. III.4: WBS do Sistema de Gerência de Projetos de Desenvolvimento de Software.

DESCRIÇÃO DAS FUNÇÕES

1. Planejamento do Projeto

1.1. Planejamento do Processo de Desenvolvimento

Tem como objetivos permitir detalhar as atividades do ciclo de vida, especificado através de XAMÃ, em unidades mais facilmente gerenciáveis e prever marcos para controle gerencial. O detalhamento deve ser refinado pelo usuário com o aumento do conhecimento à medida que se avança no desenvolvimento.

Como produto desta função é gerado o Plano do Processo de Desenvolvimento contendo as definições especificadas no modelo de PPDS descrito no Anexo I deste trabalho.

1.2. Planejamento da Gerência

Tem como objetivo apoiar o usuário na definição de uma estratégia adequada para gerenciar o projeto, levando em conta as diferenças existentes entre cada projeto, as particularidades dos diferentes ambientes de desenvolvimento e os diferentes níveis de produtividade das diversas formas de estruturas organizacionais. A definição desta estratégia implica em escolher os métodos e ferramentas a serem utilizados nas atividades de planejamento e acompanhamento do projeto, e a definição de políticas, regras e procedimentos a serem aplicados.

Para apoiar o usuário nesta escolha, o sistema oferece dois recursos: uma base de conhecimentos, contendo regras para a indicação de métodos e ferramentas, e uma base de dados de projetos passados associada a uma ferramenta para análise estatística dos dados históricos dos projetos semelhantes ao

projeto em desenvolvimento.

Como produto desta função é gerado o Plano de Gerência contendo as definições especificadas no modelo de PPDS.

1.3. Planejamento dos Recursos:

Tem como objetivos:

- estimar os recursos de pessoal (os recursos de hardware e software são especificados em XAMÃ);
- apoiar a definição de uma estrutura organizacional apropriada ao projeto e permitir a atribuição de pessoal aos níveis de responsabilidade especificados nesta estrutura;
- permitir a preparação dos cronogramas do projeto.

Como produto desta função é gerado o Plano de Recursos, contendo as estimativas e definições especificadas no modelo do PPDS.

1.4. Planejamento da gerência de riscos

Tem como objetivo apoiar a avaliação e o controle dos riscos. Para apoiar a identificação dos riscos o sistema oferece um "checklist" de riscos possíveis. Para apoio à análise do impacto e priorização dos riscos identificados é oferecido acesso a informações históricas de projetos semelhantes e uma base de conhecimentos sobre problemas e soluções. Estes recursos auxiliam, também, o controle dos riscos avaliados, permitindo a definição dos procedimentos para monitoração dos riscos e a elaboração dos planos de contingência para cada risco.

O produto desta função é o Plano de Gerência de Riscos (vide modelo do PPDS).

1.5. Planejamento da Documentação:

Tem como objetivo fornecer, para ADS's que não possuem Plano de Documentação associado, roteiros básicos de documentação para que sejam adaptados pelo usuário aos requisitos de documentação do projeto.

1.6. Planejamento do Controle da Qualidade:

Tem como objetivo permitir a definição do Plano do Controle da Qualidade. O Plano do IEEE (IEEE [71]) é oferecido como roteiro padrão.

1.7. Planejamento da Gerência da Configuração:

Tem como objetivo permitir a definição do Plano da Gerência da Configuração. O Plano do IEEE (IEEE [70]) é oferecido como roteiro padrão.

1.8. Planejamentos Adicionais:

Tem como objetivo permitir a criação de planos adicionais, como por exemplo: Plano de Treinamento, Plano de Implantação e/ou Operação, Plano para Manutenção do Produto, Plano de Gerência de Subcontratados, Plano de Segurança, etc.

2. Acompanhamento do Projeto

2.1. Controle do progresso

Tem como objetivos:

- permitir o registro diário do trabalho executado pela equipe do projeto;
- fazer a monitoração do progresso de acordo com o estabelecido

no Plano do Projeto;

- apoiar as revisões do projeto ao final de cada fase e nos marcos pré-estabelecidos, servindo de interface para a área de garantia da qualidade e para o controle da aprovação dos produtos;
- exibir o status do projeto num dado momento;
- apoiar o controle dos recursos de pessoal, financeiros, paralisações de equipamentos e problemas relacionados.

2.2. Monitoração dos Riscos

Tem como objetivos:

- apoiar a execução dos Planos de contingência, quando necessário;
- indicar, através da base de conhecimentos, soluções possíveis para problemas não previstos; e,
- permitir a simulação de ações para apoiar a tomada de decisão.

2.3. Revisão do Plano

Tem como objetivos:

- permitir alterações no Plano do Projeto durante a execução do mesmo, controlando consequências, e,
- controlar as versões de todos os documentos de gerência do projeto.

2.4. Conclusão do Projeto

Tem como objetivos apoiar o encerramento e avaliação do projeto, armazenando na base de dados históricos as informações

relevantes sobre o projeto e permitindo o refinamento da base de conhecimentos.

3. Acompanhamento do Produto

3.1. Gerência de Versões

Tem como objetivo oferecer ferramentas para a gerência de versões do produto desenvolvido.

3.2. Gerência de Modificações

Tem como objetivos gerenciar as propostas de modificação do produto, permitindo a avaliação do impacto, alocação de pessoal para executar as modificações e apoio aos demais aspectos envolvidos no processo de alterar um software.

Estas funções determinam que o sistema possua as seguintes bases de dados e de conhecimentos:

1: Base de Características de Projeto

Contém as características de projetos em diferentes domínios de aplicação.

2: Base de Métodos e Ferramentas de Gerência

Contém informações sobre métodos e ferramentas de gerência de projetos, classificados por tipos de projeto e fase de desenvolvimento.

3: Base de Conhecimento

Contém informações no formato de regras que indicam métodos e ferramentas de gerência, estruturas organizacionais e soluções para problemas típicos de projetos de desenvolvimento de software.

4: Base de Modelos de Estruturas Organizacionais

Contém modelos de estruturas organizacionais.

5: Base de Dados históricos

Contém informações relevantes de projetos passados:

- características,
- tamanho, dados de custos, duração, produtividade, métricas, observações sobre avaliações, testes, entre outras,
- métodos de gerência, políticas, regras e procedimentos aplicados,
- planos,
- recursos designados,
- ações tomadas pelo gerente do projeto, desenvolvedores, mantenedores ou geradas pelo sistema.

6: Base de Padrões de Modelos de Planos de Projeto

Contém padrões de planos de projeto de desenvolvimento.

7: Planos de Projetos

Contém os planos dos projetos em desenvolvimento.

8: Relatos de acompanhamento

Contém todos os dados de acompanhamento do projeto, incluindo os dados métricos coletados.

9: Revisões e auditorias

Contém as informações referentes às revisões e auditorias realizadas.

10: Versões

Hipertexto contendo as versões dos documentos utilizados na gerência do projeto e os motivos que levaram a cada descarte.

Como a implementação de todo o ambiente de gerência proposto excede o escopo de uma tese de mestrado, escolhemos para implementar uma de suas ferramentas, a ferramenta para planejamento de recursos. A seguir descreveremos a ferramenta implementada.

III.4) A FERRAMENTA DE PLANEJAMENTO DE RECURSOS

Como já descrito na seção anterior, as principais atividades da função de planejamento de recursos são:

- estimar os recursos de pessoal, que envolve: cálculo da estimativa de esforço, tempo, produtividade, equipe média e a distribuição do esforço e tempo pelas atividades definidas no ciclo de vida;
- definir a estrutura organizacional do projeto e atribuir responsabilidades aos níveis identificados nesta estrutura;
- preparar os cronogramas do projeto, definindo, entre outros aspectos, marcos de controle, entrega dos produtos e consumo de recursos.

A Estação TABA é uma estação configurável para diferentes domínios de aplicação, onde é possível planejar e criar o ambiente mais adequado ao desenvolvimento de um produto específico. Assim, necessita-se para o TABA de uma ferramenta de planejamento de recursos também configurável, capaz de acomodar diferentes tipos de projetos, bem como as diferenças entre os diversos ADS's gerados para cada projeto.

Para estimar os recursos necessários para o desenvolvimento de software não existem, porém, modelos de estimativa confiáveis, de aplicabilidade geral e adaptáveis a qualquer tipo

de projeto e ADS. Os modelos existentes são dependentes dos bancos de dados associados aos ambientes particulares nos quais foram desenvolvidos.

Além disto, a maioria dos modelos existentes leva a um grande problema: estimar linhas de código durante a análise de requisitos. Sabe-se que o esforço para desenvolver software dependerá cada vez menos do resultado em linhas de código, que tende a passar a ser apenas uma saída automática de uma ferramenta de especificação de requisitos, variando de acordo com a linguagem e algumas restrições de projeto. Assim sendo, o esforço estaria em especificar o sistema, independente do paradigma ou ADS, que são recursos tecnológicos usados pelo engenheiro de software para gerar o sistema.

A tendência, hoje, é a maior parte do esforço se concentrar nas primeiras fases do projeto. O engenheiro de software passa, então, a ser requisitado a conhecer o negócio tanto quanto técnicas de desenvolvimento de software, e as empresas tendem a não mais admitir especificações de sistemas que não estejam de acordo com a estratégia e o ponto de vista do negócio da empresa.

Os parâmetros de entrada para os modelos de estimativa de recursos deveriam ser algo que representasse melhor este esforço. Uma solução desta natureza esbarra, porém, na subjetividade dos aspectos relacionados a uma especificação.

Estudando as dificuldades relacionadas a estimativa de custos de software e as limitações das soluções até então propostas, optamos por, ao invés de propor uma solução teórica, direcionar nossos esforços de forma a dotar a Estação TABA com

ferramentas capazes de acomodar o atual estado da arte e atender de forma realística necessidades de estimativas presentes e futuras.

Para isto, fundamentamos nossa solução em dois aspectos: na criação de uma base de conhecimentos, onde informações colhidas de experiências passadas de modelagem são usadas para impulsionar futuros esforços, e no uso de métodos estatísticos de análise de dados aplicados a um banco de dados de projetos passados.

Com o TABA cria-se a oportunidade de se ter um repositório de projetos de variados domínios de aplicação desenvolvidos em ADS's distintos. Isto viabiliza o uso de métodos estatísticos, essencial no processo para se obter um modelo de custos realístico.

Dados empíricos podem ser interpretados inteligentemente e ter sua significância determinada usando-se ferramentas estatísticas.

Quando existe uma relação linear entre dois conjuntos de números é freqüentemente desejável expressar o relacionamento de forma analítica: seja y a variável dependente e x a variável independente, desejamos conhecer os coeficientes b_0 e b_1 tal que a curva $y = b_0 + b_1x$ seja uma boa aproximação para y .

Um método muito usado para determinar b_0 e b_1 é o método dos mínimos quadrados: b_0 e b_1 são escolhidos de forma que a soma dos quadrados dos erros $(\sum_{i=1}^n (y_i - \hat{y}_i)^2)$ seja mínima. Uma estreita

linha de aproximação obtida por mínimos quadrados é usualmente chamada de linha de regressão. É sempre possível encontrar a linha de regressão, mesmo para dados que não sejam linearmente relacionados, como é o caso dos modelos de estimativa de custos

de software, usando artifícios de linearização em sua equação.

Muitos pacotes estatísticos suportam regressão com uma ou mais variáveis. Normalmente quanto maior o número de variáveis, menor os erros quadrados esperados. A adequação de um modelo de regressão linear pode ser avaliada inspecionando o diagrama de dispersão dos resíduos, que são simplesmente a diferença entre os valores previstos e os reais observados.

Na regressão multilinear os coeficientes obtidos para variáveis independentes altamente correlacionadas não sugerem, porém, a importância relativa das variáveis na estimativa da variável dependente. A técnica para mostrar a quantidade de variância numa variável dependente explicada pelas variáveis independentes é chamada Análise de Variância (ANOVA). ANOVA nos permite ver como combinações de variáveis independentes afetam a variável dependente.

Pode-se concluir, portanto, que aplicando-se técnicas de regressão múltipla a um conjunto de dados históricos de projetos com características de aplicação e ADS semelhantes ao do projeto que se quer estimar o custo, é possível determinar a melhor curva de ajuste para uma equação de estimativa de esforço em função do tamanho, que atenda às características da aplicação e do ADS especificado, e determinar, também, que fatores de ajuste para o esforço estimado são mais relevantes para o tipo de projeto e ADS em questão.

A partir dos resultados obtidos com esses métodos estatísticos é criada uma base que permitirá o desenvolvimento de um modelo geral, capaz de suportar a diversificação imposta pelos requisitos do TABA.

A implementação da solução proposta requer alimentar a base de conhecimentos, com informações sobre avaliações de experiências de modelagem, e a base de dados históricos, com informações sobre os custos de projetos desenvolvidos na Estação. Como ponto de partida, enquanto não se dispõe dessas informações no TABA, montamos estas bases com informações retiradas da literatura pesquisada, disponibilizando a ferramenta para uso imediato.

Como foi relativamente pequena a quantidade e diversificação dos dados históricos conseguidos para entrada no processo de regressão, que tem como objetivo gerar, direta e automaticamente, modelos de estimativa adequados a variados domínios de aplicação e diferentes tipos de ADS's, foi implementada, também, uma solução paliativa, para ser utilizada enquanto não for possível dispor de informações suficientes.

Essa solução consiste da implementação de modelos existentes, da avaliação do desempenho desses modelos aplicados ao conjunto de projetos semelhantes ao projeto em questão e da indicação do modelo que obteve melhor resultado. Os modelos existentes implementados servirão tanto para se obter estimativas, como para iniciar um esforço de modelagem totalmente novo, no sentido de desenvolver um modelo geral e dinâmico, composto a partir da combinação dos melhores recursos de diversos modelos existentes e recursos de modelos futuros que se mostrem promissores.

Foi implementada, também, como parte dessa solução paliativa, uma ferramenta de calibração baseada em análise de regressão, capaz de calibrar o modelo indicado se a quantidade de projetos semelhantes for suficiente para evitar resultados

distorcidos na regressão.

Além disto, em qualquer hipótese, está disponível, também, a base de conhecimentos para indicar entre os modelos existentes o mais apropriado ao projeto em questão.

A seguir descrevemos um método que viabiliza a solução adotada.

III.4.1) Método proposto

Para estimar os custos de desenvolvimento de software, dividimos a estimativa de recursos em 4 etapas:

1) Seleção de projetos semelhantes

A seleção de projetos semelhantes tem como objetivo fornecer subsídios para as três etapas seguintes.

A seleção dos projetos semelhantes no banco de dados histórico é feita a partir do grau de influência das características considerados como as que mais riscos oferecem ao sucesso de um projeto ou que mais influenciam nos custos de projetos. Estas características terão sua relevância determinada pela ferramenta estatística implementada. Enquanto não se dispõe na base de dados históricos de informações em número suficiente para tal, partimos de um conjunto inicial obtido selecionando entre as características encontradas na bibliografia pesquisada (Boehm [4] [49], Ghezzi [17], Fairley [47], entre outros) as consideradas com maior influência nos custos e de maior risco a um projeto de software.

À cada característica definida é atribuído um número representando seu grau de influência numa escala de 0 a 5. Esta

escala foi definida baseada no intervalo adotado por modelos como COCOMO, Pontos por Função e MACOS.

Consideramos dois tipos de características como eliminatórias para seleção de projetos semelhantes:

- o tipo de aplicação, que classificamos como aplicativos, utilitários ou sistemas operacionais, e
- a tecnologia utilizada, que classificamos em batch, online, tempo real, inteligência artificial, sistemas distribuídos, ou outros.

Um projeto para ser considerado semelhante deve ser do mesmo tipo, utilizar a mesma tecnologia e possuir, ou o mesmo grau de influência, ou diferir de apenas uma unidade para cada uma das características.

Dividimos as características selecionadas em cinco categorias:

a) Características de produto:

- Integridade,
- Tamanho do banco de dados,
- Complexidade,
- Arquitetura de processamento de dados e algoritmos inovativos.

b) Características do ADS:

- Desenvolvimento concorrente de hardware e procedimentos operacionais,
- Estabilidade da plataforma,
- Disponibilidade de recursos de hardware e software,
- Disponibilidade de pessoal de suporte,
- Uso de ferramentas de software,
- Uso de práticas modernas de programação.

c) Características de pessoal:

- Experiência da equipe com ADS's semelhantes,
- Experiência da equipe com sistemas semelhantes,
- Experiência do gerente,
- Rotatividade de pessoal.

d) Características dos requisitos:

- Completeza,
- Volatilidade,
- Ambigüidade,
- Requisitos em evolução.

e) Restrições de prazo:

- Percentual do tempo de desenvolvimento normal.

Várias das características selecionadas são parâmetros de entrada para os modelos de estimativa de custo implementados na ferramenta.

2) Escolha do modelo de estimativa de custos

Nesta etapa é configurado o modelo adequado ao projeto em questão, de acordo com suas características definidas na etapa anterior. Enquanto não se dispõe de um número de informações suficiente na base de dados históricos para tal, esta etapa será realizada da forma descrita a seguir.

Como os modelos algorítmicos mais conhecidos utilizam dois tipos de abordagem, orientada a tamanho e orientada a função, nessa versão da ferramenta foram escolhidos para compor a base de modelos da Estação os dois modelos de estimativa de custos mais bem sucedidos e conhecidos, e que representam estes dois tipos de abordagem, respectivamente: o modelo COCOMO Intermediário e o

modelo analítico Pontos por Função.

Foi implementado, também, o modelo MACOS, no sentido de fazer uma avaliação de seu desempenho e posteriormente torná-lo substituto do modelo Pontos por Função, se seu desempenho for significativamente melhor. Esses três modelos se encontram descritos na Seção II.3.2 do segundo capítulo deste trabalho.

A indicação do modelo apropriado é feita através da análise estatística do desempenho dos modelos disponíveis, no caso de existir dados históricos de projetos com características semelhantes, e/ou pela base de conhecimentos.

Na avaliação de desempenho dos modelos disponíveis o critério para indicar um modelo é que este obtenha $\overline{MER} \leq 0,25$ e $PRED(0,25) \geq 0,75$, onde:

$$\overline{MER} \text{ (Magnitude Média do Erro Relativo)} = 1/n \sum_{i=1}^n MER_i,$$

$$MER \text{ (Magnitude do Erro Relativo)} = \left| \frac{(E_{real} - E_{estimado})}{E_{real}} \right|,$$

$$PRED \text{ (Predição no nível 1)} = \% \text{ da amostra que obteve } MER \leq 1$$

3) Calibração dos modelos

Para viabilizar o uso dos modelos existentes implementados é feita uma calibração usando análise de regressão, tendo o esforço como variável dependente e o tamanho como variável independente. Para evitar distorções a calibração só será realizada se existirem no banco de dados históricos pelo menos dez projetos com características semelhantes. Os modelos implementados que são calibráveis são o modelo COCOMO e o modelo MACOS. Para o modelo COCOMO a regressão é feita para os três modos (orgânico, difuso e

restrito) simultaneamente e entram para a regressão os dezesseis atributos de custo do modelo, conforme proposto por Gulezian[43] numa reformulação estatística desse modelo.

4) Cálculo das estimativas de custos

A estimativa é calculada através das equações do modelo escolhido e engloba a estimativa de esforço, tempo, produtividade e equipe média.

Para os modelos existentes implementados que não possuem equações para cálculo do tempo, produtividade e equipe média, estes dados serão obtidos pela média dos resultados de projetos históricos semelhantes.

Como a definição das atividades do ciclo de vida no TABA é livre, não é utilizada qualquer tabela de distribuição de esforço ou tempo por atividades. Apenas é oferecido apoio através da base de conhecimentos para o usuário fazer a distribuição. O conhecimento armazenado, até então, sobre esse domínio do conhecimento foi colhido na literatura pesquisada sobre estudos de fatores que afetam a produtividade.

III.4.2) A Ferramenta de Planejamento de Recursos no contexto do TABA

A ferramenta de Planejamento dos Recursos proposta atua em dois ambientes da Estação TABA: no meta-ambiente e no ambiente de desenvolvimento. No meta-ambiente o usuário recebe apoio para escolha do modelo de estimativa de custos e do tipo de estrutura organizacional adequadas ao projeto.

No ambiente de desenvolvimento a ferramenta:

- calibra o modelo de estimativa escolhido (desde que este seja calibrável);
- apóia o usuário na estimativa do tamanho do produto a ser desenvolvido, parâmetro de entrada no cálculo da estimativa de custos;
- calcula, através do modelo selecionado, a estimativa de esforço, tempo, produtividade e equipe média;
- apóia o usuário na distribuição do esforço e tempo pelas atividades especificadas no ciclo de vida do projeto;
- permite a preparação do organograma do projeto e a atribuição de responsabilidades aos níveis identificados nesta estrutura organizacional, e,
- prepara os cronogramas do projeto, de acordo com as definições do usuário.

O apoio ao usuário é dado através de:

- bases de conhecimentos, contendo regras para cada aspecto enfocado, criadas a partir dos trabalhos referenciados nesta pesquisa, e,
- uma base de dados históricos associada a uma ferramenta estatística.

III.4.3) Especificação dos requisitos

A solução proposta para a realização do planejamento de recursos é descrita a seguir através da especificação de requisitos, utilizando a Análise Estruturada (Gane [24]).

A função "Planejar Recursos" (Figura III.4) é detalhada nos diagramas das figuras III.5, III.6 e III.7. A seguir são

descritos sucintamente seus processos. No Anexo II encontra-se o Dicionário de Dados correspondente aos Diagramas de Fluxos de Dados apresentados.

Fig. III.5: DFD de Planejar Recursos (nivel 0)

Fig. III.b: DFD de Estimar custos (nível 1)

Fig. III.7: DFD de Planejar organizacao (nivel 1)

III.4.3.1) Descrição dos processos

P1 - Estimar recursos

Tem como objetivo fazer o cálculo da estimativa de custos do projeto.

P2 - Planejar a Organização

Tem como objetivo apoiar a definição da estrutura organizacional do projeto e fazer a alocação de recursos para os níveis identificados nesta estrutura.

P3 - Preparar cronogramas

Tem como objetivo preparar os cronogramas do projeto a partir das informações do usuário, das estimativas calculadas, das atividades do ciclo de vida e da alocação de recursos de pessoal, hardware e ferramentas.

P1.1 - Obter projetos semelhantes

Tem como objetivo selecionar os projetos semelhantes de acordo com o critério adotado.

P1.2 - Indicar modelo de estimativa de custos

Tem como objetivo apoiar a escolha do modelo de estimativa de custos mais adequado as características do projeto.

P1.3 - Calibrar modelo de estimativa de custos

Tem como objetivo calibrar o modelo escolhido às características do projeto.

P1.4 - Calcular estimativas

Faz o cálculo da estimativa de esforço, tempo, produtividade e equipe média através do modelo selecionado e apoia o usuário na distribuição do esforço e tempo pelas atividades especificadas no ciclo de vida. Para apoiar a estimativa de tamanho, entrada para o cálculo das estimativas, é

fornecido ao usuário informações sobre os projetos passados semelhantes. O apoio a distribuição de esforço e tempo/atividades é dado a partir da base de conhecimentos.

P2.1 - Definir estrutura organizacional

Tem como objetivo apoiar a definição do tipo de estrutura organizacional apropriada às características do projeto.

P2.2 - Alocar pessoal

Tem como objetivo permitir a atribuição de responsabilidades aos níveis identificados na estrutura organizacional.

III.4.4) Projeto de Arquitetura

O projeto de arquitetura está representado nos diagramas das figuras III.8, III.9, III.10 e III.11, apresentados a seguir.

III.4.4.1) Diagramas de estrutura modular

Fig. III.8. Estrutura modular do Sistema de Gerência

Fig. III.9. Estrutura modular de "PROCESSAR PLANEJAMENTO DOS RECURSOS"(nível 1)

Fig. III.10. Estrutura modular de "PLANEJAR ORGANIZAÇÃO" (nível 2)

Fig. III.1.1. Estrutura modular de "PROCESSAR MODELO ESCOLHIDO"(nivel 2)

III.4.4.2) Descrição dos principais módulos

OBTER CARACTERÍSTICAS DO PROJETO (fig. III.9)

Exibe janela para o usuário informar o tipo, tecnologia e grau de influência das características que servirão de base para selecionar os projetos semelhantes.

OBTER PROJETOS SEMELHANTES (fig. III.9)

Compara o tipo, a tecnologia e o grau de influência do projeto com a de cada projeto histórico selecionando como semelhante os que forem de mesmo tipo, mesma tecnologia e tiver o grau de influência de cada característica igual ou diferindo de um.

PROCESSAR MODELOS DE ESTIMATIVA (fig. III.9)

A partir dos dados dos projetos semelhantes é feita a avaliação de cada modelo implementado no sistema. Para cada projeto semelhante é estimado o esforço, calculado a partir de tamanho real através dos modelos em avaliação, e é calculada a magnitude do erro relativo.

INDICAR MODELOS DE ESTIMATIVA (fig. III.9)

Se o desempenho do modelo passou no critério adotado ele é indicado para uso. A indicação é dada, também, através da base de conhecimento, a partir das respostas dadas pelo usuário as questões formuladas pelo sistema.

OBTER MODELO ESCOLHIDO (fig. III.9)

Obtém do usuário a definição do modelo a ser utilizado daqui por diante para cálculo das estimativas. Se nenhum modelo for escolhido será assumido o modelo COCOMO.

CALIBRAR MODELO ESCOLHIDO (fig. III.11)

Passa os dados parametrizados dos projetos semelhantes para

a rotina estatística para cálculo da regressão e obtêm os coeficientes calibrados do modelo.

OBTER PARÂMETROS GERAIS (fig. III.11)

Obtem do usuário a estimativa geral de tamanho e o grau de influência dos atributos de custo do modelo em questão, exceto os já obtidos anteriormente quando da definição das características para seleção de projetos semelhantes.

OBTER DEFINIÇÃO DOS COMPONENTES (fig. III.11)

Obtém do usuário a definição dos componentes, a estimativa de tamanho de cada um e, para o caso do modelo aceitar, o grau de influência dos atributos de custo específicos do componente.

CONSULTAR PROJETOS SEMELHANTES (fig. III.11)

Exibe dados reais dos projetos históricos, tais como: características, tamanho (em MIFE, PFB e PF), esforço, tempo, produtividade (em MIFE/HM e HM/PF) e equipe média.

CALCULAR ESTIMATIVA DO MODELO (fig. III.11)

Faz o cálculo da estimativa de custos pelas equações do modelo escolhido e exibe os resultados.

DEFINIR ESTRUTURA ORGANIZACIONAL (fig. III.10)

Indica um modelo de estrutura organizacional apropriado para o projeto a partir das respostas dadas pelo usuário a questões formuladas pelo sistema e obtém do usuário para cada função que consta do organograma o nome da função, responsável e subordinados diretos e indiretos (quando for o caso).

ALOCAR PESSOAL (fig. III.10)

Obtém do usuário a definição dos cargos necessários para compor a equipe do projeto e permite que se selecione pessoas para preencher cada cargo definido. São fornecidas informações

sobre o perfil, habilidades e disponibilidade das pessoas cadastradas no sistema.

Na Seção seguinte é feita uma descrição, em forma de breve relatório, sobre a implementação da ferramenta projetada.

III.4.5) Implementação

Escopo:

A implementação contemplou o esqueleto geral do Sistema de Gerência de Desenvolvimento de Software e uma de suas ferramentas, a Ferramenta de Planejamento de Recursos.

Recursos:

Foram gastos aproximadamente 4 meses entre especificação, projeto de arquitetura e projeto detalhado, e um total de 44 dias na implementação.

Ambiente de desenvolvimento:

A ferramenta foi implementada no seguinte ambiente:

- Hardware: Estação Sun.
- Sistema operacional: Unix.
- Ambiente de desenvolvimento: Openwindow.
- Linguagens: XVIEW, C e Prolog.
- Sistema de arquivos utilizado: arquivos UNIX em formato binário com estrutura de dados tipo registro.

Reutilização de software:

Para implementar os recursos estatísticos necessários para a ferramenta proposta, optou-se desde o princípio por utilizar software pronto, evitando investir mais tempo na implementação. Era necessário, porém, acesso aos fontes desse software, para permitir integrá-lo com os demais módulos do sistema, e retirar

qualquer interface com o usuário.

Procuramos através da Internet rotinas com fontes de domínio público, não obtendo sucesso. Encontramos dois pacotes estatísticos desenvolvidos pelo Núcleo de Computação Eletrônica (NCE/UFRJ), obtendo acesso aos fontes de um desses pacotes, o PECS (Helt [69]), pacote estatístico para microcomputadores PC implementado em Fortran.

Para reutilizar a parte do código do PECS necessária para implementar os recursos estatísticos desejados, foram usados dois utilitários da SUN: um conversor de arquivos em DOS para UNIX e um conversor de programas em linguagem Fortran para linguagem C.

Ferramentas:

Foram utilizadas no desenvolvimento as seguintes ferramentas:

- gerador de interface: Guide,
- depurador de linguagem C: DBXTOOL

Montagem da Base de Dados Históricos inicial:

Como ainda não se dispõe de informações sobre projetos desenvolvidos na própria Estação Taba para alimentar a Base de Dados Históricos, foi criada uma base inicial a partir de conjuntos de dados publicados, de forma a viabilizar os testes e disponibilizar a ferramenta para uso imediato.

Foram utilizadas duas fontes: Boehm [4], que contém dados dos 63 projetos da TRW usados para criar o modelo COCOMO, e Guerra [39], que contém dados de 40 projetos, usados para criar o modelo MACOS. Foram usados, também, apenas durante os testes, dados publicados em Conte [31], Navlakha [36] e Gulezian [43].

Validação: Para validar os resultados obtidos nos diversos módulos do sistema foram usados dados publicados e comparados os

resultados, tendo o sistema obtido resultados satisfatórios.

Ambiente operacional da ferramenta:

O ambiente de operação é o mesmo ambiente utilizado no desenvolvimento.

CAPÍTULO IV

CONCLUSÃO

Este trabalho faz parte do projeto TABA (Rocha [9] Rocha [10]). O TABA visa a construção de uma estação de trabalho configurável capaz de criar um ambiente de desenvolvimento de software (ADS) adequado às necessidades de cada projeto de desenvolvimento de software.

No contexto do Projeto TABA, um ADS deve conter um conjunto integrado de métodos e ferramentas, que suportem as atividades de construção do produto, as atividades de planejamento e controle da qualidade e as atividades de gerência do processo de desenvolvimento.

Para dar suporte gerencial automatizado, imprescindível principalmente quando é grande a complexidade do sistema a ser desenvolvido, não existem, porém, ambientes capazes de suportar as diferenças existentes entre as características próprias de cada domínio de aplicação.

Esta tese, a partir da pesquisa realizada sobre gerência de projetos de desenvolvimento de software, propôs um ambiente como solução, capaz de fornecer o necessário apoio ao gerente de projetos na configuração de uma estratégia gerencial adequada ao desenvolvimento de cada projeto específico, e de permitir o acompanhamento e controle efetivos do projeto.

Foi especificado em alto nível um sistema com este objetivo, "SGPDS: Sistema de Gerência de Projetos de Desenvolvimento de Software", um sistema de apoio a decisão baseado em conhecimento.

O Sistema de Gerência proposto foi concebido como um conjunto de ferramentas integrado a uma base de conhecimentos, através do qual o gerente elaborará o Plano do Projeto e realizará o acompanhamento do processo de desenvolvimento.

Como parte desta tese foi implementada uma das ferramentas do sistema proposto, "Planejamento de Recursos", desenvolvida de forma a atender ao requisito primordial do TABA (ser configurável para diferentes domínios de aplicação), e de maneira a poder se integrar aos demais subsistemas com os quais tem interface.

A ferramenta implementado se destaca por ser um sistema de apoio a decisão baseado no conhecimento, que oferece, além de ferramentas para estimativa, organização e distribuição dos recursos, suporte para executar estas atividades da forma mais adequada às características da aplicação a ser desenvolvida e do ADS a ser utilizado.

Este suporte é dado através da indicação do modelo de estimativa e da forma de estrutura organizacional mais adequadas ao projeto, e do apoio na distribuição de recursos.

O sistema proposto se destaca, também, pelo seu suporte à decisão, que inclui bases de dados históricos e de conhecimentos, e pela sua proposta de reutilização de informações gerenciais, compiladas quando do encerramento de um projeto, possibilitando que as experiências em gerenciamento sirvam de base para o planejamento e resolução de problemas em futuros desenvolvimentos.

Além dessas facilidades, a ferramenta desenvolvida é importante, principalmente por ser, possivelmente, a primeira implementação de um ambiente com recursos de métodos estatísticos de análise de dados, capaz de não só avaliar, indicar e calibrar

automaticamente modelos de estimativa de software existentes, adaptando-os às necessidades de cada projeto e ADS, como também dar suporte interativo a novos esforços de modelagem, disponibilizando informações resultantes das análises processadas.

A ferramenta de planejamento de recursos encontra-se implementada e estará disponível para uso, junto com os demais componentes da Estação TABA.

É possível, no entanto, sua utilização desde já, independente dos demais componentes do Taba. Para isso é necessário apenas que se supra a entrada de dados naqueles arquivos acessados pela ferramenta cuja entrada de dados é feita através de outros componentes da Estação. Esses arquivos são:

- o arquivo tipo texto XAMÃ.PROJ, que contém a relação dos projetos em desenvolvimento e é alimentado através de XAMÃ, e,
- o arquivo em formato binário HISTÓRICO, que é o banco de dados históricos de projetos passados e é alimentado através do Módulo de Enceramento do Projeto, parte do Subsistema de Acompanhamento do Projeto.

O formato de ambos pode ser encontrado no arquivo fonte global.c, que faz parte do conjunto de fontes da ferramenta implementada.

A ferramenta construída implementou parcialmente o Sistema de Gerência de Projetos de Desenvolvimento de Software. A próxima versão deverá abranger as ferramentas que implementam as demais funções que completam o ambiente proposto:

- Planejamento da Gerência,
- Planejamento da Gerência de Riscos,

- Planejamento do Controle da Qualidade,
- Planejamento da Gerência da Configuração,
- Planejamento da Documentação,
- Controle do Progresso,
- Monitoração dos Riscos,
- Revisões do Plano,
- Conclusão de projetos,
- Gerência de Versões, e,
- Gerência de Modificações.

Além disso, a base de dados históricos e a base de conhecimentos deverá ser incrementada para englobar, além das informações relativas a recursos, as informações pertinentes às demais funções.

É necessário, também, para aumentar a confiabilidade das informações fornecidas pelo sistema, aumentar o número de projetos passados armazenados no Banco de Dados Históricos inicial, diversificando os tipos de aplicação e ADS's. A Base de Dados Históricos é o único fator limitante para que se possa utilizar a ferramenta na plenitude de seus objetivos e usufruir de todos os benefícios advindos do seu uso.

O Software Metrics Research Group da Universidade de Purdue mantém há vários anos uma base de dados históricos de projetos passados que vem sendo alimentada através de contatos diversos com indústrias. Sendo possível, essa é uma fonte que pode ser utilizada para ampliar a base de dados históricos inicial, enquanto não se dispõe de informações sobre projetos desenvolvidos na Estação TABA.

REFERÊNCIAS BIBLIOGRÁFICAS

1. KEANE, J.F.; KEANE, M.; TEAGAN, M. Productivity management in the development of computer applications. Prentice-Hall, 1984.
2. ROCHA, A.R.C. Análise e projeto estruturado de sistemas. Editora Campus, 1987.
3. PRESSMAN, R.S. Software engineering, a practitioner's approach. Mcgraw-Hill, 1987.
4. BOEHM, B.W. Software engineering economics. Prentice-Hall, 1981.
5. DEMARCO, T. Controle de projetos de software. Editora Campus, 1989.
6. EVANS, M.W.; MARCIANIAK, J.J. Software quality assurance and management. Wiley-Interscience, 1987.
7. PAGE-JONES, M. Gerenciamento de projetos, uma abordagem prática e estratégica no gerenciamento de projetos. McGraw-Hill, 1985.
8. JONES, C. Produtividade no desenvolvimento de software. McGraw-Hill, 1991.
9. ROCHA, A.R.C.; SOUZA, J.M. O Projeto TABA. Rio de Janeiro, COPPE/UFRJ, 1991. (Relatório Técnico do Programa de Engenharia de Sistemas e Computação, ES-145/88)
10. ROCHA, A.R.C.; SOUZA, J.M.; AGUIAR, T.C. Taba: An Heuristic Workstation for Software Development. In: COMPEURO 90, Tel Aviv, Israel, 1990. Anais. Tel Aviv, 1985. p.-.
11. RIBEIRO, Carlos Alberto Soares. Um Sistema de Apoio à Avaliação de Custos de Software. Rio de Janeiro, 1989.

- 214p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
12. MENEZES, Irene Madeira Garcia. Estime: Um Sistema para Estimar Custos de Desenvolvimento de Software. Rio de Janeiro, 1988. 183p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
13. MACHADO, José Alberto da Costa. A Medição da Produtividade no Desenvolvimento de Software. Rio de Janeiro, 1990. 253p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
14. KOBLITZ, Leonardo Falcão. Uma Ferramenta Automatizada para Gerenciamento da Configuração de Software. Rio de Janeiro, 1990. 160p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
15. VALE, Marcus Antonius de Oliveira. Prodoc: Uma Ferramenta para Produzir Documentação de Produtos de Software. Rio de Janeiro, 1988. 226p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
16. ROOK, P. Controlling Software Projects. Software Engineering Journal, v.1, n.1, p.7-16, 1986.
17. GHEZZI, C.; JAZAYERI, M.; MANDRIOLI. Fundamentals of Software Engineering. Prentice-Hall, 1991.

18. COSTA, J.L.R. Sobre a Gerência de Projetos de Software. Rio de Janeiro, 1990. 26p. Monografia (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
19. HURST, R.S. SPMMS - Information Structures in Software Management. Software Engineering Journal, v.1, n.1, p.50-57, 1986.
20. WOLVERTON, R.W. Software Costing. In: VICK, C.R.; RAMAMOORTHY, C.V. Handbook of Software Engineering. s.l., Van Nostrand Reinhold, 1984. Cap.22, p.469-493.
21. BOEHM, B.W. Software Life Cycle Factors. In: VICK, C.R.; RAMAMOORTHY, C.V. Handbook of Software Engineering. s.l., Van Nostrand Reinhold, 1984. Cap.23, p.494-518.
22. MADDY, D.B.; GRUNES, M. Design of an intelligent interface for software planning. Information and Software Technology, v.29, n.10, p.540-547, 1987.
23. AGUIAR, Tereza Cristina de. Um Sistema Especialista de Suporte a Decisão para Planejamento de Ambientes de Desenvolvimento de Software. Rio de Janeiro, 1992. 229p. Tese (Doutorado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
24. GANE, C.; SARSON, T. Análise Estruturada de Sistemas Livros Técnicos e Científicos, 1983.
25. CRISPIM, Emília. Avaliação de métodos para o desenvolvimento de software. Rio de Janeiro, 1991. 344p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
26. DUARTE, M. A. Um sistema para representação do conhecimento

- de métodos de desenvolvimento de software. Rio de Janeiro, 1991. 313p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
27. WERNECK, Vera. Taxonomia de domínios de aplicação. Rio de Janeiro, 1990. 216p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
28. CHITMAN, Edith. Estudo e classificação de modelos de ciclo de vida. Rio de Janeiro, 1991. 149p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
29. GLITZ, Selma. Aquisição do conhecimento. Rio de Janeiro, 1992. 163p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
30. ROCHA, A.R.C; SOUZA, J. The Architecture of Taba-HEP Workstation. In: International Conference on Computing in High Energy Physics '91, Tsukuba, Japão, 1991. Anais. Tsukuba, , 1991. p.-.
31. CONTE, S.D.; DUNSMORE, H.E.; SHEN, V.Y. Software Engineering Metrics and Models. The Benjamin/Cummings Publishing Company Inc., 1986.
32. ALBRECHT, A.J. Measuring Application Development Productivity. IBM System Journal, v.17, n.1, 1978.
33. CRONHJORT, B. Testing and Validation and its management. In: IFAC Workshop on Management of Software Projects, 3., West Lafayette, Indiana, 1989. Anais. Purdue University, West Lafayette, Indiana, 1989.

34. PULK, B.E. Improving Software Project Management. Journal of Systems and Software, n.13, p.231-235, 1990.
35. VERNER, J.; TATE, G. A Model for Software Sizing. Journal of Systems and Software, n.7, p.173-177, 1987.
36. NAVLAKHA, J.K. Choosing a Software Cost Estimation Model for Your Organization: A Case Study. Information and Management, n.18, p.255-261, 1990.
37. HO, Y.; MCDEVITT, C.D. Determination of Optimal Resource Allocation for Software Development - An Application of A Software Equation. Information and Management, n.18, p.79-85, 1990
38. KUSTERS, R.J.; GENUCHTEN, M.V.; HEEMSTRA, F.J. Are Software Cost-Estimation models Accurate?. Information and Software Technology, v.32, n.3, p.187-190, 1990.
39. GUERRA, L.; LOBOS, H. An Empirical Validation of a Software Cost Model. In: IFAC Workshop on Management of Software Projects, 3., West Lafayette, Indiana, 1989. Anáís. Purdue University, West Lafayette, Indiana, 1989.
40. HELMER, O. Social Technology. Basic Books, 1966.
41. BASILI, V.R. Resource Models. Tutorial on models and metrics for software management and engineering, IEEE Computer Society Press, p.4-9, 1980.
42. DEMARCO, T. An Algorithm for sizing software products. Performance Evaluation Review, n.12, p.13-22, 1984.
43. GULEZIAN, R. Reformulating and Calibrating COCOMO. Journal of Systems Software, n.16, p.235-242, 1991.
44. KEMERER, C.F. An empirical validation of software cost models. Communications of the ACM, v.30, n.5, p.416-429,

1987.

45. MOHANTY, S.N. Software cost estimation: present and future. Software Practice Experience, v.11, 1981.
46. BROOKS, F.P. The Mythical Man-Month. Addison-Wesley, Reading, 1975.
47. FAIRLEY, R.E. Risk Management: the key to sucessful software projects. In: IFAC Workshop on Management of Software Projects, 3., West Lafayette, Indiana, 1989. Anáís. Purdue University, West Lafayette, Indiana, 1989.
48. AIR FORCE SYSTEMS COMMAND. Software Risk Management. AFSC Pamphlet 800-45, 1987.
49. BOEHM, B. Tutorial Notes on Software Risk Management of Software. In: Conferência Internacional em Engenharia de Software, 10., Singapura, 1988. Anáís. Singapura, 1988.
50. DEFENSE SYSTEMS MANAGEMENT COLLEGE. Risk Assessment Techniques. Fort Belvoir, Virginia, 1983.
51. NATIONAL SECURITY INDUSTRY ASSOCIATION. Proceedings Software Risk Management. 1987.
52. WIDEMAN, R. Risk Management. Project Management Journal, v.17, n.4, 1986.
53. ROCHA, A.R.C.; AGUIAR, T.C.; BLASCHEK, J. Ambientes para desenvolvimento de software: definição de termos. Rio de Janeiro, COPPE/UFRJ, 1987. (Relatório Técnico do Programa de Engenharia de Sistemas e Computação, ES-137/87).
54. Modelo Padrão do IEEE para Planos de Gerência de Projetos de Software (versão Preliminar).
55. BOEHM, B.W.; ROSS, R. Theory-W Software Project Management: Principles and Examples. IEEE Transactions on Software Engineering, v.15, n.7, 1989.

56. EVANS, M.W.; PIAZZA, P.; DOLKAS, B. Principles of Productive Software Management. Wiley, 1983.
57. METZGER, P.W. Managing a Programming Project. 2. ed., Prentice-Hall, 1981.
58. BOEHM, B.W.; ROSS, R. Theory-W Software Project Management: A Case Study. IEEE Transactions on Software Engineering, p30-40, 1988."
59. ABDEL-HAMID, T.K.; MADNICK, S.E. Lessons Learned from Modeling the Dynamics of Software Development. Communications of the ACM, v.32, n.12, p1426-1437, 1989.
60. ABDEL-HAMID, T.K.; MADNICK, S.E. Impact of Schedule Estimation on Software Project Behavior. IEEE Software, p70-75, 1986.
61. DIXON, D. Integrated Support for Project Management. IEEE Software, p.49-58, 1988.
62. ROCHA, A.R.C. et al. An environment for Software Quality Evaluation. In: International Conference on Quality Assurance, Jerusalém, Israel, 1992. Anáís. Jerusalém, Israel, 1992.
63. MOURA, Lúgia. Taxonomia de Ambientes de Desenvolvimento. Rio de Janeiro, 1992. 186p. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro.
64. PRESMAN, R.S. Software Engineering: a practitioner's approach. 3. ed. New York, Mcgraw-Hill, 1992.
65. FERNANDES, A.A.; KUGLER, J.L.C. Gerência de Projetos de Sistemas, uma abordagem prática. Livros Técnicos e Científicos Editora Ltda, 1989.

66. YOURDON, E. Análise Estruturada Moderna. Editora Campus, 1990.
67. FEIGENBAUM, E.; MCCORDUCK, P. The fifth generation. Reading, Mass, Addison-Wesley, 1983.
68. LEDERER, A.L.; PRASAD, J. Nine management guidelines for better cost estimating. Communications of the ACM, v.35, n.2, p51-59, 1993.
69. HELT, S.R.M.; BOMFIM, M.N.C.; MARTINS, R.C. Estatística em microcomputadores - Uma nova opção para usuários de SPSS. Livros Técnicos e Científicos Editora S.A., 1985.
70. IEEE 828-1983.
71. IEEE 730-1984.
72. SAVAFI, A. Computer-Based Risk Management In Software Project Planning/Scheduling. In: IFAC Workshop on Management of Software Projects, 3., West Lafayette, Indiana, 1989. Análs. Purdue University, West Lafayette, Indiana, 1989.
73. LUCONI, F.L.; MALONE, T.W.; MORTON, M.S.S. Expert Systems: The Next Challenge for Managers. Sloan Management Review, p1-13, 1986.
74. HART, A. Knowledge-acquisition for expert systems. New York, Mcgraw-Hill, 1986.
75. WIELINGA, B.J. et al. Knowledge Acquisition for Expert Systems. In: ACAI, 1988. Análs. 1988.

ANEXO I

MODELO PROPOSTO DE PLANO DO PROJETO (PPDS)

Página de Título

Sumário

Prefácio

Índice

Lista de Figuras

Lista de Tabelas

1. Introdução

2. Visão Geral do Projeto

3. Plano do Processo de Desenvolvimento

4. Plano da Gerência do projeto

5. Plano de Recursos

6. Plano da Gerência de Riscos

7. Plano de Documentação

8. Plano da Gerência da Configuração

9. Plano do Controle da Qualidade

10. Planos Adicionais (opcional)

Apêndices (opcional)

DESCRIÇÃO DOS COMPONENTES DO PPDS

1. Introdução

1.1 Propósito

(Objetivos do Plano do Projeto de Desenvolvimento de Software (PPDS))

1.2 Audiência

(Pessoas a quem se destina o PPDS)

1.3 Visão Geral do Plano do Projeto

(Como está organizado o PPDS)

1.4 Material Referenciado

(Lista dos documentos e outras fontes referenciadas no plano, como por exemplo arquivos eletrônicos. Para os documentos devem constar: título, número (no caso de relatórios técnicos) e órgão de publicação. Para outras fontes devem constar: número da versão e data. Qualquer desvio dos padrões ou políticas referenciadas devem ser identificadas nesta seção.)

1.5 Definição de Termos e Abreviaturas

(Definições diretas ou referenciadas de termos e abreviaturas necessários para interpretar o plano apropriadamente.)

1.6 Plano de Controle de alterações no PPDS

(Procedimentos de alterações e mecanismos de controle abrangendo os procedimentos para alteração do plano (procedimentos pré-estabelecidos no cronograma e procedimentos não previstos), procedimentos para disseminar as alterações e controle de versões do plano (mecanismo para colocar a versão inicial sob controle de alterações e para controlar alterações

subseqüentes).)

2. Visão Geral do Projeto

2.1 Sumário do Produto

(Visão geral do produto a ser entregue.)

2.2 Objetivos do Projeto

(Visão geral do projeto com uma lista de todos os itens a serem entregues, datas de entrega, locais de entrega e quantidade requerida para satisfazer os termos do acordo do projeto. Esta lista de produtos do projeto não deve ser feita como estabelecimento oficial dos requisitos do projeto.)

2.3 Entidades envolvidas

(Todas as entidades externas a serem envolvidas no projeto.)

2.4 Principais atividades, produtos e marcos

(Atividades do projeto reunidas em grandes grupos, produtos gerados por cada uma e os marcos que os separam.)

2.5 Recursos requeridos

(Lista dos recursos requeridos pelo projeto abrangendo hardware, métodos, ferramentas, equipe média, recursos financeiros e cronograma geral.)

3. Plano do Processo de Desenvolvimento

3.1 Modelo do processo

(Modelo gráfico e textual do ciclo de vida mostrando o relacionamento entre as atividades, como especificadas pela escolha dos produtos, marcos principais, baselines, revisões e

aprovações que ocorrem pelo ciclo de vida.)

3.2 Hardware, métodos e ferramentas do Processo de desenvolvimento

(Identificação especificando hardware, metodologias de desenvolvimento, linguagem de programação, ferramentas, técnicas e métodos e a serem usados para especificar, projetar, construir, testar, integrar, documentar, liberar, modificar e/ou manter os produtos do projeto. Além disto, padrões técnicos, políticas e procedimentos que governam o processo de desenvolvimento e/ou de modificação destes produtos.)

3.3 Prioridades relativas entre os requisitos

(Lista dos requisitos em ordem de prioridade.)

3.4 Suposições, dependências e restrições

(Estabelecimento das suposições nas quais o projeto está baseado, eventos externos dos quais o projeto é dependente e as restrições sob as quais o projeto será conduzido.)

3.5 Relacionamento com outros projetos

(Relacionamento deste projeto com outros projetos, se apropriado.)

4. Plano da Gerência do Projeto

4.1 Estratégia Gerencial

4.1.1 Detalhamento das atividades

(Detalhamento das atividades do ciclo em unidades facilmente gerenciáveis.)

4.1.2 Detalhamento dos recursos das atividades gerenciais:

(Ferramentas, métodos, técnicas e estratégias

gerenciais (Políticas, procedimentos, regras, prioridades gerenciais, etc) a serem empregados pela gerência, tanto para o planejamento como para o acompanhamento.)

4.2 Plano do acompanhamento do projeto

4.2.1 Mecanismos para monitoração e controle do projeto

(Mecanismos para relato, formato e frequência dos relatórios, fluxo das informações, mecanismos de revisões e auditoria e relacionamento destes mecanismos com a gerência da configuração e o controle da qualidade.)

5. Plano de Recursos

5.1 Organização do Projeto

(Estrutura organizacional (organograma), interfaces organizacionais (limites administrativos e gerenciais entre o projeto e outras organizações), organizações externas (organização superior, do cliente, dos usuários, de subcontratados, etc), organizações de suporte (controle da qualidade, gerência da configuração, etc).)

5.2 Recursos Humanos

(Recursos técnicos, dando para cada um dos níveis técnicos requeridos a identificação, quantidade, data de início, duração e responsável, apoio administrativo necessário, métodos para obter, treinar, reter e liberar e matriz de responsabilidades.)

5.3 Outros recursos

(Qualquer recursos necessário distinto dos já mencionados.)

5.4 Custos

(Dados relativos aos custos do projeto englobando as

estimativas de esforço, tempo, produtividade, equipe média, custo, distribuição dos recursos por fases e por atividades, e cronogramas do projeto.)

6. Plano da Gerência de Riscos

6.1 Riscos identificados

(Lista de todos os riscos identificados do projeto, considerando-se riscos contratuais, riscos tecnológicos, riscos devido ao tamanho e complexidade do produto, riscos na aquisição e retenção de pessoal, e riscos na obtenção da aceitação do produto pelo cliente.)

6.2 Análise dos riscos

(Análise do impacto potencial em termos de prejuízo de cada risco identificado.)

6.3 Priorização dos riscos

(Lista dos riscos identificados priorizados em ordem de impacto.)

6.4 Plano de contingência para cada risco

(Planos com ações emergenciais para recuperação dos riscos.)

7. Plano de Documentação

7.1 Requisitos de Documentação

(Lista de todos os documentos requeridos como parte dos requisitos do projeto.)

7.2 Guia de estilo

(Normas e roteiros para a documentação requisitada, referenciados ou diretamente incluídos nesta seção.)

7.3 Convenções de nomes

(Convenções dos nomes adotados nos documentos, referenciados ou diretamente incluídos nesta seção.)

7.4 Formato dos Documentos

(Formato de documentos gerados pelas atividades do projeto, referenciados ou diretamente incluídos nesta seção.)

7.5 Marcos

(Marcos de entrega dos documentos requeridos.)

7.6 Revisões

(Datas, participantes e diretrizes gerais para as revisões dos documentos a liberar.)

7.7 Aprovações

(Datas, responsáveis e requisitos para as aprovações dos documentos liberados.)

8. Plano da Gerência da Configuração do Software

(Modelo do Plano de Gerência da Configuração do IEEE, diretamente ou referenciado:

Modelo IEEE de PGCS (IEEE 828-1983)

Seções:

Seção 1 - Introdução

1.1 - Propósito

1.2 - Escopo

1.3 - Definições e siglas

1.4 - Referências

Seção 2 - Gerenciamento

2.1 - Organização

2.2 - Responsabilidades

2.3 - Controle de Interface

2.4 - Implementação do PGCS

2.5 - Políticas, Diretrizes e Procedimentos Aplicáveis

Seção 3 - Atividades do Gerenciamento da Configuração do Software

3.1 - Identificação da Configuração

3.2 - Controle da Configuração

3.3 - Contabilidade da Situação da Configuração

3.4 - Auditorias e Revisões

Seção 4 - Ferramentas, Técnicas e Metodologias

Seção 5 - Controle de Fornecedor (contratado)

Seção 6 - Retenção e Coleção de Registros

9. Plano do Controle da Qualidade

(Modelo do Plano de Controle da Qualidade do IEEE, diretamente ou referenciado:

Modelo IEEE de PCQS (IEEE 730-1984)

Seções:

Seção 1 - Propósito

Seção 2 - Documentos Referenciados

Seção 3 - Gerenciamento

3.1 - Organização

3.2 - Tarefas

3.2 - Responsabilidades

Seção 4 - Documentação

4.1 - Propósito

4.2 - Requisitos Mínimos de Documentação

4.2.1 - Especificação dos Requisitos

4.2.2 - Descrição do Projeto

4.2.3 - Plano de Verificação e Validação

4.2.4 - Relatório de Verificação e Validação

4.2.5 - Documentação do Usuário

4.3 - Outros

Seção 5 - Padrões, Práticas e Convenções

5.1 - Propósito

5.2 - Conteúdo

Seção 6 - Revisões e Auditorias

6.1 - Propósito

6.2 - Requisitos Mínimos

6.2.1 - Revisão dos Requisitos

6.2.2 - Revisão Preliminar do Projeto

6.2.3 - Revisão Crítica do Projeto

6.2.4 - Revisão de Verificação e Validação

6.2.5 - Auditoria Funcional

6.2.6 - Auditoria Física

6.2.7 - Auditoria no Processo

6.2.8 - Revisões Gerenciais

Seção 7 - Gerência da Configuração do Software

Seção 8 - Relatório de Problema e Ação Corretiva

Seção 9 - Ferramentas, Técnicas e Metodologias

Seção 10 - Controle do Código

Seção 11 - Controle do Meio

Seção 12 - Controle de Fornecedor (contratado)

Seção 13 - Coleção, Manutenção e Retenção de Registros)

10. Planos Adicionais

(Planos opcionais, que podem ser necessários, e que devem ser incluídos em seções e subseções adicionais ao PPDS. Podem ser:

- Plano de treinamento;
- Plano de Instalação/Operação (conversão dos dados, transição de sistema, operação, etc);
- Plano para Manutenção do Produto;
- Plano de Gerência de Subcontratados;
- Plano para Garantia da Segurança, etc.)

ANEXO II

Descrição detalhada dos processos

P1 - Estimar recursos

Descrição: Tem como objetivo fazer o cálculo da estimativa de custos do projeto.

Entradas:

a-P1 - Características do projeto

D2-P1 - Projetos passados

D4-P1 - Conhecimento de modelos

a-P1 - Modelo escolhido

a-P1 - Estimativas de tamanho

D3-P1 - Ciclo de vida

D4-P1 - Conhecimento distr. esforço

Saídas:

P1-a - Indicação de modelo

P1-a - Projetos semelhantes

P1-a - Estimativas de custos

P1-D1 - Dados da estimativa de custos

P2 - Planejar a Organização

Descrição: Tem como objetivo apoiar a definição da estrutura organizacional do projeto e fazer a alocação de recursos para os níveis identificados nesta estrutura.

Entradas:

a-P2 - Definição da estrutura organizacional

D4-P2 - Conhecimento sobre estruturas organizacionais

D5-P2 - Dados de pessoal

a-P2 - Registro de alocação

Saídas:

P2-a - indicação de estrutura organizacional

P2-a - Pessoal disponível

P2-D1 - Dados da organização

P3 - Preparar cronogramas

Descrição: Tem como objetivo preparar os cronogramas do projeto a partir das informações do usuário, das estimativas calculadas, das atividades do ciclo de vida e da alocação de recursos de pessoal, hardware e ferramentas.

Entradas:

D1-P3 - Dados de distribuição de recursos

a-P3 - Definição dos cronogramas

D3-P3 - Ciclo de vida, hardware e ferramentas

Saídas:

P3-D1 - Cronogramas definidos

P3-a - Cronogramas

P1.1 - Obter projetos semelhantes

Descrição: Tem como objetivo selecionar os projetos semelhantes de acordo com o critério adotado.

Entradas:

a-P1.1 - Características do projeto

D2-P1.1 - Projetos passados

Saídas:

P1.1-D1/1 - Registro de projetos semelhantes

P1.1-D1 - Características informadas

P1.2 - Indicar modelo de estimativa de custos

Descrição: Tem como objetivo apoiar a escolha do modelo de estimativa de custos mais adequado as características do projeto.

Entradas:

D1/1-P1.2 - Dados históricos para avaliar modelos

D4-P1.2 - Conhecimento de modelos de estimativa de custos

a-P1.2 - Modelo escolhido

Saídas:

P1.2-a - Indicação de modelo

P1.2-D1 - Modelo para cálculo

P1.3 - Calibrar modelo de estimativa de custos

Descrição: Tem como objetivo calibrar o modelo escolhido às características do projeto.

Entradas:

D1/1-P1.3 - Dados históricos para calibrar modelo

a-P1.3 - Pedido para calibrar

Saídas:

P1.3-D1 - parâmetros do modelo calibrado

P1.4 - Calcular estimativas

Descrição: Faz o cálculo da estimativa de esforço, tempo, produtividade e equipe média através do modelo selecionado e apóia o usuário na distribuição do esforço e tempo pelas atividades especificadas no ciclo de vida. Para apoiar a estimativa de tamanho, entrada para o cálculo das estimativas, é fornecido ao usuário informações sobre os projetos passados semelhantes. O apoio à distribuição de esforço e tempo/atividades é dado a partir da base de conhecimentos.

Entradas:

a-P1.4 - Dados para estimativas

D1-P1.4 - Parâmetros de cálculo

D3-P1.4 - Ciclo de vida

D4-P1.4 - Conhecimento distr. esforço

D1/1-P1.4 - Tamanho real de projetos

Saídas:

P1.4-a - Tamanho de projetos semelhantes

P1.4-D1 - Dados da estimativa de custos

P1.2-a - Estimativa de recursos

P2.1 - Definir estrutura organizacional

Descrição: Tem como objetivo apoiar a definição do tipo de estrutura organizacional apropriada às características do projeto.

Entradas:

a-P2.1 - Definição da estrutura organizacional

D4-P2.1 - Conhecimento sobre estruturas organizacionais

Saídas:

P2.1-a - Indicação estrutura organizacional

P2.1-D1 - Dados da estrutura organizacional

P2.2 - Alocar pessoal

Descrição: Tem como objetivo permitir a atribuição de responsabilidades aos níveis identificados na estrutura organizacional.

Entradas:

D5-P2.2 - Dados de pessoal

a-P2.2 - Registro de alocação

Saídas:

P2.2-a - Pessoal disponível

P2.2-D1 - Dados da alocação

Dicionário de dados

1) Entidades externas

1.1) Gerente do projeto (a)

- Descrição: Pessoa responsável pelo planejamento e acompanhamento do projeto para quem o sistema gera o Plano de Recursos.

- Fluxo de dados que entram:

P1-a, P1.2-a - Indicação de modelo

P1-a, P1.4-a - Tamanho de projetos semelhantes

P1-a, P1.4-a - Estimativas de custos

P2-a, P2.1-a - indicação de estrutura organizacional

P2-a, P2.2-a - Pessoal disponível

P3-a - Cronogramas

- Fluxo de dados que saem:

a-P1, a-P1.1 - Características do projeto

a-P1, a-P1.2 - Modelo escolhido

a-P1.3 - Pedido de calibração

a-P1, a-P1.4 - Dados da estimativa

a-P2, a-P2.1 - Definição da estrutura organizacional

a-P2, a-P2.2 - Registro de alocação

a-P3 - Definição dos cronogramas

2) Depósitos de dados

2.1) Plano de recursos (D1)

- Descrição: contém as informações relativas à estimativa de recursos, à organização e aos cronogramas para os projetos em desenvolvimento.

- Fluxo de dados que entram:

P1-D1 - Dados da estimativa de custos

P1.1-D1 - Características informadas

P1.2-D1 - Modelo para cálculo

P1.3-D1 - Parâmetros do modelo calibrado

P1.4-D1 - Dados da estimativa de custos

P2.1-D1 - Dados da organização

P2.1-D1 - Dados da estrutura organizacional

P2.2-D1 - Dados de alocação

P3-D1 - Cronogramas definidos.

- Fluxo de dados que saem:

D1-P1.4 - Parâmetros de cálculo

D1-P3 - Dados de distr. de recursos

- Conteúdo: Plano-de-recursos

2.2) Projetos semelhantes (D1/1)

- Descrição: contém as informações relativas aos projetos semelhantes.

- Fluxo de dados que entram:

P1.1-D1 - Registro de projetos semelhantes

- Fluxo de dados que saem:

D1/1-P1.2 - Dados históricos para avaliar modelos

D1/1-P1.3 - Dados históricos para calibrar modelos

D1/1-P1.4 - Tamanho real de projetos

- Conteúdo: Registro-projeto-passado

2.3) Banco de dados histórico (D2)

- Descrição: contém as informações relativas aos projetos já encerrados.

- Fluxo de dados que saem:

D2-P1,D2-P1.1 - Projetos passados

- Conteúdo: Registro-projeto-passado

2.4) ADS (D3)

- Descrição: contém as informações relativas ao ADS definido através de XAMÃ.

- Fluxo de dados que saem:

D3-P1, D3-P1.4, D3-P3 - Ciclo de vida

- Conteúdo: Especificação-ads

2.5) Base de conhecimentos (D4)

- Descrição: contém informações no formato de regra que indicam, de acordo com as características do projeto, modelos de estimativa de custos, tipos de estruturas organizacionais e como fazer a distribuição do esforço e tempo por atividades.

- Fluxo de dados que saem:

D4-P1,D4-P1.2 - Conhecimento de modelos de estimativa de custos

D4-P1,D4-P1.4 - Conhecimento sobre distribuição de esforço por atividades

D4-P2,D4-P2.1 - Conhecimento sobre estruturas organizacionais

2.6) Pessoal (D5)

- Descrição: contém informações sobre pessoal disponível para alocação.

- Fluxo de dados que saem:
- D5-P2, D5-P2.2 - Dados de pessoal
- Conteúdo: Registro-pessoal

3) Fluxos de dados

3.1) a-P1, a-P1.1 - Características do projeto

- Descrição: Graus de influência das características do projeto, informados pelo usuário para seleção de projetos passados semelhantes.

- Conteúdo: {características-do-projeto}

3.2) P1-D1, P1.1-D1 - Características informadas

- Descrição: Graus de influência das características do projeto a serem registrados.

- Conteúdo: {características-do-projeto}

3.3) D2-P1, D2-P1.1 - Projetos passados

- Descrição: Informações sobre projetos passados.

- Conteúdo: {características-do-projeto} + {dados-de-tamanho} + {dados-reais-de-custos}

3.4) P1.1-D1/1 - Projetos semelhantes

- Descrição: Informações sobre projetos passados semelhantes, que servirão de entrada para os processos P1.2, P1.3 e P1.4.

- Conteúdo: {características-do-projeto} + {dados-de-tamanho} + {dados-reais-de-custos}

3.5) D4-P1.2 - Conhecimento de modelos

- Descrição: Regras para indicação de modelos de estimativa de custos.

3.6) P1-a, P1.2-a - Indicação de modelo

- Descrição: Nome(s) de modelo(s) indicado(s) para o usuário pelo sistema.

- Conteúdo: {nome-de-modelo}*

3.7) a-P1, P1.2 - Modelo escolhido

- Descrição: Nome do modelo de estimativa de custos escolhido pelo usuário.

- Conteúdo: {nome-de-modelo}

3.8) P1.2-D1 - Modelo para cálculo

- Descrição: Nome do modelo escolhido para cálculo da estimativa de custos, para ser armazenado.

- Conteúdo:

3.9) D1/1-P1.2 - Dados históricos para avaliar modelos

- Descrição: Dados de projetos semelhantes para cálculo da estimativa de esforço através dos os modelos a serem avaliados.

- Conteúdo: {{atributos-de-custo-do-modelo} +
 {tamanho-real-na-unidade-do-modelo} +
 {esforço-real}}*

3.10) a-P1.3 - Pedido para calibrar

- Descrição: Pedido do usuário ao sistema para calibrar o modelo escolhido.

- Conteúdo: {nome-de-modelo}

3.11) D1/1-P1.3 - Dados históricos para calibrar modelo

- Descrição: Dados de projetos semelhantes para calibrar o modelo escolhido.

- Conteúdo: {{atributos-de-custo-do-modelo} +
 {tamanho-real-na-unidade-do-modelo} +
 {esforço-real}}*

3.12) P1.3-D1 - Parâmetros do modelo calibrado

- Descrição: Coeficientes calibrados do modelo escolhido.
- Conteúdo: {coeficientes-do-modelo-calibrados}

3.13) a-P1, P1.4 - Dados para estimativas

- Descrição: Estimativa de tamanho geral, ou por componentes, e atributos de custo do modelo não incluídos nas características informadas para determinar os projetos semelhantes. .ls1

- Conteúdo:{{tamanho-geral} + {atributos-complementares}}
{{dados-e-tamanho-do-componente}+ {atributos}}*

3.14) P1.4-D1 - Dados calculados da estimativa de custos

- Descrição: Dados de custo calculados pelo modelo escolhido para serem armazenados.

- Conteúdo: {estimativa-de-custo}

3.15) P1-D1 - Dados da estimativa de custos

- Descrição: Todos os dados relativos a estimativa de custo a serem armazenados.

- Conteúdo: {estimativa-de-custo}

3.16) P1.4-a - Estimativa de custos

- Descrição: Dados de custo calculados pelo modelo escolhido, fornecidos ao usuário.

- Conteúdo: {estimativa-de-custo}

3.17) P1.4-a - Tamanho de projetos semelhantes

- Descrição: Tamanho real geral de projetos semelhantes e por componentes.

- Conteúdo: {tamanho-real-geral} +
{identificação-e-tamanho-do-componente}*

3.18) D1-P1.4 - Parâmetros de cálculo

- Descrição: Atributos de custo para o modelo escolhido já informado na definição das características para seleção de projetos semelhantes.

- Conteúdo: {características-do-projeto}

3.19) D3-P1.4 - Ciclo de vida

- Descrição: Descrição das atividades definidas no ciclo de vida através de XAMÃ.

- Conteúdo: {dados-da-atividade}*

3.20) D4-P1.4 - Conhecimento sobre distribuição de esforço e tempo por atividades

- Descrição: Regras para indicar, de acordo com as características do projeto e do ADS, como deve ser distribuído o esforço pelas atividades.

3.21) a-P2, a-P2.1 - Definição da estrutura organizacional

- Descrição: Dados para definir a estrutura organizacional.

- Conteúdo: {dados-da-estrutura-organizacional}

3.22) D4-P2, D4-P2.1 - Conhecimento sobre estruturas organizacionais

- Descrição: Regras que indicam tipos de estruturas organizacionais.

3.23) P2-a, P2.1-a - indicação de estrutura organizacional

- Descrição: Tipos de estruturas organizacionais adequada as características do projeto.

- Conteúdo: {tipo-de-estrutura-organizacional}*

3.24) P2-D1 - Dados da organização

- Descrição: Todos os dados relativos ao planejamento da organização, a serem armazenados.

- Conteúdo: {Dados-da-organização}

3.25) P2.1-D1 - Dados da estrutura organizacional

- Descrição: Dados de definição da estrutura organizacional a serem armazenados.

- Conteúdo: {dados-da-estrutura-organizacional}

3.26) D5-P2, D5-P2.2 - Dados de pessoal

- Descrição: Informação sobre pessoal para compor a equipe de projeto.

- Conteúdo: {Informações-sobre-pessoal}*

3.27) a-P2.2 - Dados de alocação

- Descrição: Dados sobre a alocação de pessoal para a equipe do projeto para serem armazenados.

- Conteúdo: {Dados-de-alocação-de-pessoal}*

3.28) P2.2-D1 - Registro de alocação

- Descrição: Pedido feito pelo gerente para alocação de pessoal para a equipe do projeto.

- Conteúdo: {nome-pessoa-a-alocar}*

3.29) D1-P3 - Dados para cronogramas

- Descrição: Dados de distribuição de recursos

- Conteúdo: {identificacao-e-alocação-do-recurso}*

3.30) a-P3 - Definição de cronogramas

- Descrição: Dados necessários a definição dos cronogramas do projeto.

- Conteúdo: {definição-dos-cronogramas}

3.31) D3-P3 - Ciclo de vida, hardware e ferramentas

- Descrição: Dados sobre as atividades e os recursos de hardware e ferramentas alocados.

- Conteúdo: {dados-de-atividades} + {dados-de-hardware} +
{dados-de-ferramentas}

3.32) P3-D1 - Cronogramas definidos

- Descrição: Dados de definição dos cronogramas para serem armazenados.

- Conteúdo: {Dados-de-definição-dos-cronogramas}

3.33) P3-a - Cronogramas

- Descrição: Cronogramas definidos para o usuário.

- Conteúdo: {Dados-de-definição-dos-cronogramas}

ANEXO III

MANUAL DO USUÁRIO

Este manual tem como objetivo fornecer as informações necessárias para operar a Ferramenta de Planejamento de Recursos implementada. Para isto, está dividido em 2 partes. A primeira parte dá uma visão geral do sistema e contém os requisitos básicos para fazer o sistema funcionar. A segunda parte mostra todas as funções do sistema, explicando o objetivo de cada uma e a maneira de operá-las.

I. Planejamento de Recursos no TABA

I.1 Visão geral

As principais atividades da função de planejamento de recursos são:

- estimar os recursos de pessoal, que envolve: cálculo da estimativa de esforço, tempo, produtividade, equipe média e a distribuição do esforço e tempo pelas atividades definidas no ciclo de vida;
- definir a estrutura organizacional do projeto e atribuir responsabilidades aos níveis identificados nesta estrutura;
- preparar os cronogramas do projeto, definindo, entre outros aspectos, marcos de controle, entrega dos produtos e consumo de recursos.

A Estação TABA é uma estação configurável para diferentes domínios de aplicação, capaz de criar um ambiente adequado ao

desenvolvimento de cada projeto específico. Dessa forma, a principal característica da Ferramenta de Planejamento de Recursos é ser configurável para diferentes domínios de aplicação e diferentes tipos de ADS's.

Esta característica está presente no meta-ambiente da Estação, que indica o modelo para estimativa de custos e o tipo de estrutura organizacional mais adequados às características do produto a ser desenvolvido e do ambiente configurado para esse desenvolvimento.

I.2. Requisitos para executar o sistema

- **Hardware:** Estação Sun.
- **Ambiente operacional:** Unix/Openwindow.

II. Executando o sistema

II.1. Começando

A ferramenta de Planejamento de recursos é uma das ferramentas pertencentes ao **Subsistema de Planejamento** do Sistema de Gerência de Projetos de Desenvolvimento de Software. Para ter acesso ao sistema digite jprincipal em uma seção de comandos, como no exemplo abaixo:

```
CARAJAS% jprincipal
```

Este comando dará acesso à janela principal do Sistema de Gerência de Projetos de Desenvolvimento de Software (fig. A.III.1). Nesta janela abra o menu-botão **Planejamento** e selecione a opção **Recursos** para ter acesso à janela principal da Ferramenta de Planejamento de Recursos (fig. A.III.2).

II.2. Selecionando o projeto

Para ter acesso às funções do planejamento de recursos a primeira providência é selecionar o projeto para o qual se deseja planejar os recursos. Feito isto, o sistema ativará os botões que acionam as funções da ferramenta. Estes botões têm as seguintes funções:

O botão (**Estimativas...**) dá acesso ao Módulo de Estimativa de Recursos, exibindo sua primeira janela, onde deve ser informadas as características do projeto.

O botão (**Organização...**) dá acesso ao módulo de Planejamento da Organização, exibindo sua primeira janela, onde deverá ser definida a estrutura organizacional do projeto.

O botão (**Cronogramas...**) dá acesso ao Módulo de Preparação dos Cronogramas, exibindo sua primeira janela. Nesta versão esta opção não está implementada devido a falta de recursos de software gráfico, atualmente em desenvolvimento.

O botão (**Imprime...**) gera o Plano de Recursos no formato de relatório e exibe-o numa janela de edição de texto (Fig. A.III.22), permitindo que o usuário possa complementá-lo como desejar.

O botão (**Salva**) grava o Plano de Recursos em disco. Este é o único botão que salva o Plano em disco. Os demais botões (**Salva**) apenas salvam os dados para memória. Portanto, não se deve esquecer de acioná-lo ao fazer alterações no Plano.

II.3. Módulo de Estimativa de Recursos

Acionado através do botão (**Estimativas...**) (fig. A.III.2), está dividido nas seguintes etapas:

II.3.1) Definição das características do projeto

A janela para definição das características do projeto é a primeira janela do Módulo de Estimativa de Recursos (Fig. A.III.3). Nela o usuário deve informar, selecionando entre as opções apresentadas, o tipo, a tecnologia e o grau de influência das características orientadas a riscos que servirão para a seleção dos projetos semelhantes.

Nesta janela encontram-se dois botões:

O botão (**Salva**) grava na memória as informações alteradas nesta janela.

O botão (**Calcula estimativas...**) dá acesso à janela seguinte do Módulo de Estimativa de Recursos. Este acesso é feito de duas maneiras:

- quando o Plano de Recursos está sendo criado pela primeira vez, este botão levará o usuário para a janela de definição do modelo de estimativa a ser utilizado no projeto;
- quando o Plano de Recursos já existe, isto é, já foi salvo em disco anteriormente, este botão levará o usuário para a janela de estimativa geral de tamanho correspondente ao modelo de estimativa de custos escolhido para o projeto.

Exemplo: Suponha o projeto X com as seguintes características:

a) Características de produto:

- Integridade alta => grau de influência (g.i.) = 3,
- Tamanho do banco de dados em bytes / número de instruções fontes entregues menor que 10 => g.i. = 1,
- Complexidade muito alta => g.i. = 4,
- Necessidade de arquitetura de processamento de dados e

algoritmos inovativos extra alta => g.i.= 5,

b) Características do ADS:

- Necessidade intensa de desenvolvimento concorrente de hardware e procedimentos operacionais => g.i. = 3,
- Estabilidade da plataforma baixa => g.i. = 1,
- Disponibilidade de recursos de hardware e software alta => g.i. = 3,
- Disponibilidade de pessoal de suporte alta => g.i. = 3,
- Uso de ferramentas de software baixo => g.i. = 1,
- Uso de práticas modernas de programação baixo => g.i. = 1,

c) Características de pessoal:

- Experiência da equipe com ADS's semelhantes muito baixa => g.i. = 1,
- Experiência da equipe com sistemas semelhantes considerável => g.i. = 0,
- Experiência do gerente alta => g.i. = 3,
- Rotatividade de pessoal alta => g.i. = 3,

d) Características dos requisitos:

- Completeza alta => g.i. = 3,
- Volatilidade muito alta => g.i. = 4,
- Ambigüidade baixa => g.i. = 1,
- Requisitos em evolução normal => g.i. = 2,

e) Restrições de prazo:

- Percentual do tempo de desenvolvimento normal baixo => g.i.= 1.

II.3.2. Definição do modelo de estimativa

Nesta janela o usuário fará a definição do modelo de estimativa a ser utilizado no projeto (Fig. A.III.4). Para isto o

sistema oferece apoio através de dois recursos, acionados pelos botões **(Análise estatística)** e **(Apoio...)**.

O primeiro, análise estatística, executa a avaliação do desempenho dos modelos disponíveis na Estação a partir dos dados de projetos semelhantes, e indica os modelos que obtiveram desempenho satisfatório. Para isto, o sistema calcula para cada projeto semelhante a estimativa de esforço através de cada modelo implementado e compara os resultados calculados com o esforço real registrado, obtendo a magnitude do erro relativo. Ao final, para cada modelo é calculada a magnitude média do erro relativo e a predição no nível 0,25, verificando se o modelo satisfaz as condições exigidas ($\overline{MRE} \leq 0,25$ e $PRED(0,25) \geq 0,75\%$) para que seja indicado.

O segundo botão, Apoio, dá acesso ao sistema de apoio à decisão baseado em conhecimento implementado que indica um modelo para a estimativa de custos, conforme descrito a seguir.

II.3.3. Apoio

Através desta janela (fig. A.III.5) o usuário será questionado sobre uma série de fatores e, de acordo com suas respostas, o sistema indicará o modelo de estimativa mais adequado.

Exemplo: ver fig. A.III.5.

II.3.4. Estimativa do tamanho geral do produto a ser desenvolvido

A janela correspondente à estimativa geral de tamanho varia de acordo com o modelo escolhido (fig. A.III.6, A.III.7 e A.III.8). Nela deve ser informado o tamanho geral do produto (na

unidade correspondente ao modelo escolhido), o valor médio de 1 Homem-mês e os atributos gerais do modelo. Alguns dos atributos do modelo escolhido podem não aparecer nesta janela por já terem sido informados como parte das características para seleção de projetos semelhantes.

Nestas janelas encontram-se os seguintes botões:

Botão (Dados hist...) - dá acesso à janela de projetos semelhantes, onde o usuário terá apoio de dados históricos para estimar o tamanho.

Botão (Componentes...) - dá acesso a janela para definição de componentes, permitindo ao usuário a opção de realizar estimativas botton-up.

Botão (Calibra) - este botão só aparece nas janelas dos modelos calibráveis. Seu acionamento executa a rotina de regressão linear múltipla que calibrará os coeficientes do modelo escolhido a partir dos dados dos projetos semelhantes.

Botão (Custos) - efetua o cálculo das estimativas de esforço, tempo, produtividade e equipe média exibindo os resultados na janela correspondentes ao modelo escolhido. Para o cálculo são utilizadas as equações do modelo escolhido, calibradas ou não. No caso de terem sido especificados componentes são fornecidas, também, as estimativas por componentes.

Exemplo: Supondo ser o modelo COCOMO o indicado e adotado para o Projeto X temos:

Tamanho em milhares de instruções fontes entregues: 22

Características (complementando as necessárias ao modelo):

a) Características do ADS:

- Restrições de tempo de execução extra grande => g.i. = 5,
- Tempo de resposta normal => g.i. = 2,
- Restrições de memória extra grande => g.i. = 5,

b) Características do Pessoal:

- Capacidade do analista muito alta => g.i. = 4,
- Experiência do analista com a aplicação alta => g.i. = 3,
- Capacidade do programador normal => g.i. = 2,
- Experiência com a máquina virtual muito baixa => g.i. = 0,
- Experiência com a linguagem de programação muito baixa => g.i. = 1,
- Uso de práticas modernas de programação baixo => g.i. = 1,

c) Características dos Requisitos:

- Necessidade total de conformidade com os requisitos => g.i. = 2,
- Necessidade total de conformidade com especificações de interface externa => g.i. = 2,
- Entendimento geral dos objetivos do projeto => g.i. = 2.

II.3.5. Estimativa do tamanho dos componentes

A janela correspondente à estimativa do tamanho dos componentes também varia de acordo com o modelo escolhido (fig. A.III.9, A.III.10 e A.III.11). Nesta janela o usuário deve criar os componentes nos quais deseja particionar o produto, informando para cada um o nome, objetivo, tamanho (na unidade correspondente ao modelo escolhido), o valor médio de 1 Homem-mês para o componente e os atributos do componente, quando for o caso. Para definir componentes o usuário pode se utilizar dos seguintes botões:

Botão (Cria) - abre o campo Nome para se criar um componente. Nas demais ações este campo estará desabilitado para alteração. Ao terminar de entrar com as informações acione o botão salva para gravar na memória as informações fornecidas.

Botão (Salva) - salva um componente recém criado e alterações feitas para o componente selecionado na lista.

Botão (Exclui) - exclui da lista o componente selecionado.

Além destes botões encontram-se, também, nesta janela os seguintes botões:

Botão (Dados hist...), que dá acesso à janela de projetos semelhantes, onde o usuário terá apoio de dados históricos para estimar o tamanho.

Botão (Calcula Total), que auxilia o usuário fornecendo a cada instante o tamanho total do produto. Este resultado é exibido em sua janela correspondente, que encontra-se aberta neste momento.

II.3.6. Resultados da estimativa de custos

Os resultados calculados para cada modelo compreendem o tamanho, o esforço, o tempo, a produtividade, a equipe média e o custo. No caso de se ter definido componentes é calculado para cada um o tamanho, o esforço, a produtividade e o custo. Para os modelos que não possuem equações de estimativa de tempo, produtividade e equipe média, são fornecidos o valor médio dos projetos semelhantes.

A janela onde são exibidos os resultados da estimativa de custos possui apenas uma pequena variação entre os modelos (fig. A.III.12, A.III.13 e A.III.14), que consiste no fato do usuário poder alterar os valores correspondentes ao tempo, produtividade

e equipe média, quando estes tiverem sido calculados pela média dos projetos semelhantes. Para isto, as janelas que possuem esta opção (fig. A.III.13 e A.III.14) contém o botão (**Salva**), que grava na memória as alterações feitas.

O botão (**Distr. ativ...**) dá acesso à janela para fazer a distribuição do esforço e tempo pelas atividades especificadas no ciclo de vida.

Exemplo: Para o Projeto X dos exemplos anteriores teríamos:

- Esforço (HM) = 869 HM;
- Tempo (meses) = 21,8;
- Produtividade (MIFE/HM) = 0,0253;
- Equipe média = 39,86.

II.3.7. Distribuição do esforço e tempo por atividades

Nesta versão esta implementado nesta janela (fig. A.III.15) apenas o acesso à janela de apoio, uma vez que o arquivo com o ciclo de vida especificado em XAMÃ, que é entrada para esta função, vai ser em breve reformulado, com a implantação do editor gráfico que está sendo desenvolvido. Só após serão implementados os recursos para esta função.

O botão (**Apoio...**) dá acesso à janela de apoio (fig. A.III.5), já descrita anteriormente na Seção II.3.3. desse Anexo.

II.3.8. Projetos Semelhantes

As janelas apresentadas nas figuras A.III.16, A.III.17 e A.III.18 exibem os dados de projetos semelhantes.

II.4. Módulo de Planejamento da Organização

O planejamento da organização está dividido no sistema em duas fases. A primeira é a escolha e definição da estrutura organizacional a ser adotada no projeto. Para isto o usuário recebe apoio do sistema. A segunda fase é a alocação de pessoal às posições identificadas na estrutura organizacional.

II.4.1. Definição da estrutura organizacional

Na janela de definição da estrutura organizacional (fig. A.III.19) o usuário deve definir as funções identificadas fornecendo o nome da função, o responsável, a lista das funções subordinadas diretamente e, quando se tratar de estrutura matricial, a lista das funções subordinadas indiretamente. Para criar a lista de subordinados são utilizados os seguintes botões:

Botão (Cria) - abre o campo **Função** para se definir na lista um subordinado. Nas demais ações este campo estará desabilitado para alteração. Ao terminar de entrar com as informações acione o botão salva para gravação na memória.

Botão (Salva) - salva uma função definida como subordinada.

Botão (Exclui) - exclui da lista o subordinado selecionado.

Além destes botões encontram-se, também, nesta janela os seguintes botões:

O botão **(Apoio...)** dá o acesso à janela de apoio (fig. A.III.5), já descrita na Seção II.3.3..

O botão **(Prox. função)** salta para a função seguinte de mesmo nível hierárquico, como se pode ver no exemplo a seguir:

O botão (**Prox. nível**) salta para a primeira função subordinada ao nível hierárquico corrente, como se pode ver no exemplo abaixo:

O botão (**Nível ant.**) salta para a função hierarquicamente superior à corrente, seguindo o caminho inverso do percorrido através do botão anterior.

O botão (**Recursos**) dá acesso à janela da etapa seguinte do planejamento da organização, a alocação de pessoal.

II.4.1. Alocação de pessoal

Nestas janelas (fig. A.III.20 e A.III.21) é feita a alocação do pessoal para compor a equipe do projeto. Para isto são identificados os cargos necessários às responsabilidades identificadas na estrutura organizacional. Na criação da lista de cargos (janela da fig. A.III.20) são utilizados os seguintes botões:

Botão (**Cria**) - abre o campo **Cargo** para defini-lo na lista de cargos. Nas demais ações este campo estará desabilitado para alteração. Ao terminar de entrar com as informações acione o

botão salva para gravação na memória.

Botão (Salva) - salva as informações de um cargo definido e do pessoal alocado para ele.

Botão (Exclui) - exclui da lista o cargo selecionado e as pessoas alocadas a ele.

Botão (Alocação) - dá acesso à janela para alocação de pessoal ao cargo selecionado na lista.

Na janela da fig. A.III.21 são exibidas as informações relativas ao pessoal disponível para alocação. Para alocar uma pessoa selecione seu nome na lista e acione o botão (Alocar).

Fig. A.III.1

Fig. A.III.2

Características do projeto :	
Tipo: ☒ sistema operac.	Tecnologia: ☒ batch
Produto:	Pessoal:
Integridade_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Experiencia da equipe c/ ADS_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
Tamanho BD_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Experiencia equipe sistemas semelh.: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
Complexidade_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Experiencia do gerente_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
Arq. P.D. e algorit. inovativos.: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Rotatividade de pessoal_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
ADS:	Requisitos:
Desenv. concorrente HW e P.O.: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Completeza_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
Estabilidade da plataforma____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Volatilidade_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
Disponibilidade de HW e SW___: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Ambiguidade_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
Disponibilidade de suporte____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Requisitos em evolucao_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
Uso de ferramentas de SW_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	Restricoes prazo:
Praticas moder. programacao: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>	% tempo desenvolv. normal_____: 1 <input type="button" value="▲"/> <input type="button" value="▼"/>
<input type="button" value="Salva"/>	<input type="button" value="Calcula estimativas..."/>

Fig. A.III.3

Indicacao do modelo de estimativa de custos

Modelo escolhido: ☒ COCOMO

Fig. A.III.4

Fig. A.III.5

COCOMO: Estimativa do tamanho do produto em MIFE

Tamanho MIFE: 22
Valor HM: 1

Características:

ADS: *

Tempo execucao.....: 5
Tempo de resposta.....: 2
Memoria.....: 5

Requisitos:

Conformidade requisitos....: 2
Conformidade interface.....: 2
Entendimento objetivos.....: 2

Pessoal: *

Capacidade do analista.....: 4
Experiencia analista aplicacao: 3
Capacidade programador.....: 2
Experiencia maquina virtual....: 0
Experiencia linguagem progra.: 0

Restricoes de prazo:

Recompensa adiantamento: 2

(*) So p/ estimativas gerais

Dados hist...

Componentes...

Calibra

Custos...

Fig. A.III.6

MACOS: Estimativa do tamanho do produto em PFB

E: 18 ▲▼ S: 23 ▲▼ C: 33 ▲▼ I: 34 ▲▼ A: 27 ▲▼

Valor HM: 0 ▲▼

Características:

Produto:

Volume de transacoes.....: 3 ▲▼
Cap. val., atu. e arm. dados.....: 3 ▲▼
Facilidade de uso.....: 3 ▲▼
Flexibilidade de alteracao.....: 4 ▲▼
Performance requerida.....: 4 ▲▼
Facilidade de instal. e conver.: 3 ▲▼

Pessoal:

Experiencia analista c/ aplicac: 1 ▲▼
Capacidade do analista.....: 1 ▲▼
Capacidade do programador.....: 1 ▲▼
Experiencia do programador.....: 1 ▲▼
Experiencia c/ maquina virtual: 1 ▲▼
Experiencia linguagem progra.: 1 ▲▼
Experiencia do usuario.....: 1 ▲▼
Comunicacao c/ usuario.....: 3 ▲▼

ADS:

Tempo de resposta.....: 1 ▲▼
Memoria.....: 1 ▲▼
Influencia do ADS.....: 1 ▲▼
Modo de desenvolvim. e uso...: 1 ▲▼

Projeto:

Qualidade da especificacao.....: 1 ▲▼
Plano de recursos e controle....: 1 ▲▼
Planos detalhados.....: 1 ▲▼
Planos teste, V. & V. detalhados: 1 ▲▼
Seguranca e privacidade.....: 1 ▲▼
Documentacao area aplicacao...: 1 ▲▼

Dados hist...

Componentes...

Calibra

Custos...

Fig. A.III.7

Pontos por Funcao: Estimativa do tamanho do produto

Es: 1	Em: 0	Ec: 0
Ss: 2	Sm: 0	Sc: 0
Cs: 0	Cm: 0	Cc: 0
Is: 0	Im: 0	Ic: 0
As: 1	Am: 0	Ac: 0

Valor HM: 0

Caracteristicas:

Produto:

Comunicacao de dados...: 0	Atualizacao online.....: 1
Funcoes distribuidas...: 1	Processamento complexo.....: 1
Performance requerida...: 0	Reutilizabilidade.....: 1
Carga da configuracao...: 1	Facilidade de instal. e conver.: 0
Volume de transacoes...: 0	Facilidade de uso.....: 0
Entrada de dados online: 1	Multiplicidade de locais.....: 1
Eficiencia do usuario...: 1	Flexibilidade de alteracao.....: 0

Dados hist... Componentes... Custos...

Fig. A.III.8

COCOMO: Estimativa do tamanho dos componentes em MIFE

Componentes:

c1
c2

Cria salva exclui

Nome: c1

Objetivo: xxx

Tamanho MIFE: 10 ▲▼

Valor HM: 10 ▲▼

Dados hist... Calcula total

Características:

Integridade.....	1	▲▼
Tamanho BD.....	0	▲▼
Complexidade.....	1	▲▼
Tempo execucao.....	0	▲▼
Tempo de resposta.....	0	▲▼
Memoria.....	0	▲▼
Estabilidade da plataforma.....	0	▲▼
Praticas modernas programac..	0	▲▼
Uso de ferramentas de SW.....	0	▲▼
Capacidade do analista.....	1	▲▼
Experiencia analista aplicacao:	0	▲▼
Capacidade programador.....	0	▲▼
Experiencia maquina virtual....	0	▲▼
Experiencia linguagem progra:	0	▲▼
% tempo desenvolvim. normal:	0	▲▼
Votatilidade dos requisitos.....	0	▲▼

Fig. A.III.9

MACOS: Estimativa do tamanho dos componentes em PFB

Componentes:

macos 1
macos 2
comp 3
macos 4
macos 5

Entradas...: 10 ▲▼

Saidas.....: 15 ▲▼

Consultas: 15 ▲▼

Interfaces: 20 ▲▼

Arquivos...: 10 ▲▼

Cria **Salva** **Exclui**

Nome: macos 4

Objetivo: obj 4

Valor HM: 50 ▲▼

Dados hist.. **Calcula total**

Fig. A.III.10

Pontos por Funcao: Estimativa de tamanho dos componentes

Componentes:	Es: 0 <input type="text"/>	Em: 0 <input type="text"/>	Ec: 0 <input type="text"/>
pf 1	Ss: 0 <input type="text"/>	Sm: 0 <input type="text"/>	Sc: 0 <input type="text"/>
pf 2	Cs: 1 <input type="text"/>	Cm: 2 <input type="text"/>	Cc: 0 <input type="text"/>
pf 3	Is: 0 <input type="text"/>	Im: 0 <input type="text"/>	Ic: 0 <input type="text"/>
	As: 1 <input type="text"/>	Am: 0 <input type="text"/>	Ac: 0 <input type="text"/>

Características :

Comunicacao de dados...: 0 <input type="text"/>	Atualizacao online...: 0 <input type="text"/>
Funcoes distribuidas...: 0 <input type="text"/>	Processamento complexo...: 1 <input type="text"/>
Performance requerida...: 2 <input type="text"/>	Reutilizabilidade...: 0 <input type="text"/>
Carga da configuracao...: 1 <input type="text"/>	Facilidade de instal. e conver: 1 <input type="text"/>
Volume de transacoes...: 1 <input type="text"/>	Facilidade de uso...: 1 <input type="text"/>
Entrada de dados online: 0 <input type="text"/>	Multiplicidade de locais...: 1 <input type="text"/>
Eficiencia do usuario...: 0 <input type="text"/>	Flexibilidade de alteracao...: 0 <input type="text"/>

Nome: pf 1

Objetivo: obj 1

Valor HM: 10

Fig. A.III.11

COCOMO: Estimativa de custos

Tamanho (MIFE): 20 Produtividade (MIFE/HM): 0.08

Esforço (HM): 243.66 Equipe media: 16.79

Tempo (meses): 14.51 Custo: 1608.80

Componentes:

1

2

Componente:

Tamanho (MIFE): 10

Esforço HM: 140.18

Produtividade: 0.07

Custo: 1401.83

Distr. ativ....

Fig. A.III.12

MACOS: Estimativa de custos

Tamanho (PFB): 135

Tempo (meses): 5

Esforço (HM): 20.94

Equipe media: 5

Produtividade (PFB/HM): 6.45

Custo: 803.62

Salva

Componentes:

macos 1

macos 2

comp 3

macos 4

macos 5

Componente:

Tamanho (PFB): 19

Esforço HM: 2.95

Produtividade (PFB/HM): 6.44

Custo: 88.50

Distr. ativ....

Fig. A.III.13

Pontos por Funcao: Estimativa de custos

Tamanho (PF): 10.22

Esforco (HM): 70.00

Custo: 900.00

Produtividade (HM/PF): 5 ▲▼

Equipe media: 5 ▲▼

Tempo (meses): 5 ▲▼

Componentes:

pf 1

pf 2

pf 3

Componente:

Tamanho (PF): 4.00

Produtividade (HM/PF): 5 ▲▼

Esforco (HM): 20.00

Custo: 400.00

Distr. ativ....

Salva

Fig. A.III.14

Fig. A.III.15

Projetos semelhantes

Projetos semelhantes:

01	
02	
03	
04	
05	
06	
07	
08	
09	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Características...

Dados de custos...

Fig. A.III.16

Características do projeto semelhante:	
Integridade: 1	Modo desenv. e uso: 0
Tamanho BD: 4	Experiencia c/ ADS em geral: 0
Complexidade: 0	Capacidade dos analistas: 1
Arq. P.D. e algoritmos inovativos: 2	Experiencia analistas aplicacao: 1
Volume de transacoes: 0	Capacidade dos programadores: 1
Cap. val. atu. e arm. dados: 0	Experiencia programadores aplicacao: 0
Facilidade uso: 0	Experiencia c/ maquina virtual: 1
Flexibilidade alteracao: 0	Experiencia linguagem programacao: 2
Performance requerida: 0	Entendimento objetivos: 2
Facilidade instal. e convercao: 0	Experiencia do gerente: 0
Comunicacao de dados: 0	Rotatividade de pessoal: 0
Funcoes distribuidas: 0	Comunicacao c/ usuario: 0
Carga da configuracao: 0	Experiencia do usuario: 0
Entrada de dados online: 0	Completeza: 0
Eficiencia do usuario: 0	Volatilidade: 3
Atualizacao online: 0	Ambiguidade: 0
Processamento complexo: 0	Requisitos em evolucao: 0
Reutilizabilidade: 0	Conformidade c/ requisitos: 2
Multiplicidade de locais: 0	Conformidade c/ Interface: 2
Desenv. concorrente HW e P.O. 2	Qualidade final da especificacao: 0
Estabilidade plataforma: 3	Plano de recursos e controle projeto: 0
Uso ferramentas SW: 1	Planos detalhados: 0
Programacao moderna: 0	Teste, V. e V. detalhados: 0
Tempo de execucao: 2	Documentacao na area de aplicacao: 0
Tempo de resposta: 3	Seguranca e privacidade: 0
Memoria principal: 3	% tempo desenv. normal: 3
Disponibilidade suporte: 0	Recompensa: 2
Disponibilidade de HW e SW: 0	
Influencia do ambiente: 0	

Fig. A.III.17

Dados de custo																													
Tamanho																													
Esforço HM	: 2040.00	MIFE: 113					PF: 0.00	PFB: 0.00																					
Tempo meses	: 0.00	ES: 0	EM: 0	EC: 0	E: 0																								
Produtividade MIFE	: 0.00	SS: 0	SM: 0	SC: 0	S: 0																								
	PF: 0.00	CS: 0	CM: 0	CC: 0	C: 0																								
Equipe media	: 0.00	IS: 0	IM: 0	IC: 0	I: 0																								
		AS: 0	AM: 0	AC: 0	A: 0																								
<div style="display: flex; align-items: flex-start;"> <div style="width: 25%; border: 1px solid black; height: 150px; margin-right: 5px;"> <div style="position: absolute; top: 0; right: 0; width: 20px; height: 20px; border: 1px solid black; display: flex; flex-direction: column; align-items: center; justify-content: center;"> <div style="width: 10px; height: 10px; border: 1px solid black; margin-bottom: 2px;"></div> <div style="width: 10px; height: 10px; border: 1px solid black; margin-bottom: 2px;"></div> <div style="width: 10px; height: 10px; border: 1px solid black;"></div> </div> </div> <div> <p>Componentes:</p> <p>Esforço HM: 0</p> <p style="text-align: right; margin-top: 10px;">Tamanho</p> <p>MIFE: 0</p> <p>PF: 0</p> <p>PFB: 0</p> <table style="width: 100%; margin-top: 10px;"> <tr> <td>ES: 0</td><td>EM: 0</td><td>EC: 0</td><td>E: 0</td></tr> <tr> <td>SS: 0</td><td>SM: 0</td><td>SC: 0</td><td>S: 0</td></tr> <tr> <td>CS: 0</td><td>CM: 0</td><td>CC: 0</td><td>C: 0</td></tr> <tr> <td>IS: 0</td><td>IM: 0</td><td>IC: 0</td><td>I: 0</td></tr> <tr> <td>AS: 0</td><td>AM: 0</td><td>AC: 0</td><td>A: 0</td></tr> </table> </div> </div>										ES: 0	EM: 0	EC: 0	E: 0	SS: 0	SM: 0	SC: 0	S: 0	CS: 0	CM: 0	CC: 0	C: 0	IS: 0	IM: 0	IC: 0	I: 0	AS: 0	AM: 0	AC: 0	A: 0
ES: 0	EM: 0	EC: 0	E: 0																										
SS: 0	SM: 0	SC: 0	S: 0																										
CS: 0	CM: 0	CC: 0	C: 0																										
IS: 0	IM: 0	IC: 0	I: 0																										
AS: 0	AM: 0	AC: 0	A: 0																										

Fig. A.III.18

Definicao da Estrutura Organizacional

Nivel: 0

Funcao: f0

Responsavel: xxxccx

Funcoes subordinadas:

f1
f2
f3

Cria Salva Exclui

Subordinados Indiretos:

l1
l2
l3
l4

Cria Salva Exclui

Funcao: f2

Funcao:

Apolo... Prox. funcao Prox. nivel Nivel anterior Recursos...

Fig. A.III.19

Recursos Humanos requeridos

Cargo: Cargo 2
Responsavel: ???
Qtde: 3
Habilidades: ???

Cargo:

Cargo 1

Cargo 2

Cargo 3

Pessoal alocado:

Rogerio da Silveira

Pedro Carlos

Cria
Salva
Exclui
Alocacao
Salva
Exclui

Alocado:

Nome: Rogerio da Silveira
Habilidades: Digitador
Data inicio da alocacao: 09/09/93
Duracao: 20
Unid. (D/M/A): D
Status: Disponivel
Obs:

Fig. A.III.20

Recursos Humanos

Pessoal:

Jorge Marcos
Ciclano de Almeida
Pedro Carlos
Manuel Jose
Fernando Azevedo

▲

▼

Nome: Manuel Jose

Habilidades: skill 4

Status: Alocado

Alocar

Fig. A.III.21

9

PLANO DE RECURSOS

PLANO DE RECURSOS

PROJETO : teste2

1. ORGANIZACAO DAS PESSOAS

1.1 Estrutura organizacional:

Anexo 1 (organograma)

2 RECURSOS HUMANOS

2.1 Niveis tecnicos requeridos:

2.4 Matriz de responsabilidades:

Anexo 2

3. CUSTOS

3.1 Metodo escolhido:

PONTOS POR FUNCAO

Equacao utilizada:

$HM = (PF * 5.00) / 130$

3.3 Dados de custo

Tamanho total:

3.00

Unidade:

PF

Fig. A.III.22