PLANEJAMENTO DE CUSTOS EM AMBIENTES DE DESENVOLVIMENTO DE *SOFTWARE* ORIENTADOS À ORGANIZAÇÃO

Monalessa Perini Barcellos

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

_____

Profa. Ana Regina Cavalcanti da Rocha, D. Sc.

_____

Prof. Guilherme Horta Travassos, D. Sc.

_____

Profa. Káthia Marçal de Oliveira, D. Sc.

_____

Prof. Ricardo de Almeida Falbo, D. Sc.

RIO DE JANEIRO, RJ-BRASIL
JUNHO DE 2003

BARCELLOS, MONALESSA PERINI

Planejamento de Custos em Ambientes de Desenvolvimento de Software Orientados à Organização [Rio de Janeiro] 2003

VIII, 208 p., 29,7 cm (COPPE/UFRJ, M. Sc., Engenharia de Sistemas e Computação, 2003)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Planejamento de Custos

2. Ambientes de Desenvolvimento de *Software* Orientados à Organização

I. COPPE/UFRJ II. Título (série)

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc)

PLANEJAMENTO DE CUSTOS EM AMBIENTES DE DESENVOLVIMENTO DE *SOFTWARE* ORIENTADOS À ORGANIZAÇÃO

Monalessa Perini Barcellos

Junho/2003

Orientadores: Ana Regina Cavalcanti da Rocha
Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

A indústria de *software* vem produzindo *software*s cada vez mais complexos e maiores, com exigência de tempo e custos cada vez menores e com necessidade de qualidade cada vez mais acurada. Entregar um produto com qualidade, dentro do prazo e custos esperados é hoje um grande desafio para as organizações. Fatores como esses impulsionam o interesse, por parte das organizações, pela estimativa e controle dos custos dos projetos de *software*. Nas últimas décadas muitas pesquisas têm sido realizadas no sentido de desenvolver modelos para estimar custos que resultem em estimativas o mais próximo possível dos custos reais dos projetos. A utilização dos conceitos e práticas de gerência do conhecimento tem se mostrado eficiente no apoio ao planejamento de custos de projetos. Para apoiar o processo de gerência do conhecimento em organizações que desenvolvem *software*, surgiu o conceito dos Ambientes de Desenvolvimento de *Software* Orientados à Organização, que apoiam a atividade de Engenharia de *Software* em uma organização, fornecendo o conhecimento acumulado e relevante para esta atividade, assim como dando suporte ao aprendizado organizacional em Engenharia de *Software*.

Este trabalho apresenta uma abordagem para Planejamento de Custos de projetos nesses ambientes, baseada nas normas NBR ISO 10006, ISO/IEC DTR 16326, no guia PMBOK (*Project Management Body of Knowledge*), nos princípios da gerência do conhecimento e nos modelos paramétricos COCOMO II e Análise de Pontos de Função.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requeriments for the degree of Master of Science (M. Sc)

COSTS PLANNING IN ENTERPRISE-ORIENTED SOFTWARE DEVELOPMENT ENVIROMENT

Monalessa Perini Barcellos

June/2003

Advisors: Ana Regina Cavalcanti da Rocha
Guilherme Horta Travassos

Departament: Systems and Computing Engineering

Software industry must produce software more complex and larger in less time and costs reduced no loss quality. Delivering a quality software product with expected cost and on time is today a great challenge for the organizations. These issues increase organizations' interest in software projects costs estimation and control techniques. During the last decades a lot of researches have been accomplished aiming the development of cost estimation models to produce estimations close a possible to projects' real cost. Beside these techniques, Knowledge Management concepts have demonstrated to be efficient to support cost planning. Enterprise-Oriented Software Development Environment support the Software Engineering activities in an organization, providing the accumulated and considerable knowledge for these activities and supporting the learning organization in Software Engineering.

This work presents an approach for projects Costs Planning in Enterprise-Oriented Software Development Environment, based on standards such as NBR ISO 10006, ISO/IEC DTR 16326, in the guide PMBOK (Project Management Body of Knowledge), in concepts of Knowledge Management and parametric models COCOMO II and Function Point Analysis.

Ao meu pai, meu herói.

À minha filha, Luiza, o grande amor da minha vida.

Ao meu esposo e companheiro Alex Sandro.

Agradecimentos

Ao meu pai, pelo início de tudo... Por ter me ensinado o verdadeiro valor do estudo e do conhecimento.

À minha filha, Luiza, por ter vindo ao mundo durante o desenvolvimento deste trabalho mostrando-me como limites podem ser superados.

À minha filha, Luiza, mais uma vez, por me presentear com seus lindos sorrisos, carinhos e brincadeiras nos momentos de cansaço na caminhada deste trabalho.

Ao meu esposo, Alex Sandro, por mais do que qualquer pessoa, acreditar em meu potencial, oferecer o ombro sempre que precisei e por vibrar comigo a cada etapa concluída.

À minha irmã, Débora, pela companhia, cumplicidade e ajuda nos momentos mais difíceis.

À amiga Luciana, pela amizade, carinho, paciência, incentivo, troca de conhecimentos e pela força e companheirismo de todos os momentos.

Ao Sávio, por seu empenho e esforço, sem os quais este trabalho não teria se concretizado.

Ao Gleison, Karina, Lilian e Sômulo pelo auxílio com seus conhecimentos e por sua boa vontade em colaborar sempre (mesmo que nas madrugadas).

À Catarina por me acolher durante as viagens.

Ao amigo Francisco Rapchan pelo incentivo, apoio espiritual e colaboração com sua experiência e conhecimento.

Ao Blaschek pela oportunidade de trabalhar sob sua gerência e pelo carinho e apoio no primeiro ano deste trabalho.

Aos participantes da pesquisa que colaboraram prontamente à minha solicitação.

Aos meus orientadores pelos ensinamentos, direcionamentos e pela paciência nessa jornada.

Aos professores membros da banca por prestigiarem meu trabalho com sua presença.

À Ana Paula e funcionários administrativos pela prestatividade.

À CAPES pelo apoio financeiro.

A Deus... pela vida... e por este momento!

Conteúdo

Capítulo 1

Introdução

Este capítulo apresenta as principais questões que motivaram a realização deste trabalho, o objetivo da pesquisa e a organização da Tese.

1.1 Motivação

As ferramentas e técnicas utilizadas no desenvolvimento de novas aplicações de *software* não têm sido suficientes para garantir o sucesso dos projetos. Problemas significativos ainda têm sido relatados. Observa-se, por exemplo, não aderência de *baselines*[1], não cumprimento de orçamentos, elaboração de estimativas incorretas ou incoerentes e um número praticamente inaceitável de projetos cancelados, estagnados ou que não tenham atendido às expectativas dos clientes.

Tendo a modernidade tecnológica garantido a evolução dos recursos de apoio ao desenvolvimento de sistemas, como explicar tal fato? Talvez não seja tão difícil revelar e perceber que tal fato se dá pois muitos praticantes utilizam a tecnologia disponível, mas pouco conhecem os processos que ela apóia. Com isso, dificilmente seriam capazes de executar os processos sem a tecnologia de apoio, pois apenas utilizam o processo da tecnologia em si e não o processo genérico no qual a tecnologia se baseou para ser capaz de prover apoio automatizado (MURCH, 2000). Esse fato pode contribuir com os problemas anteriormente citados.

Para os executivos de negócio, o número de projetos sem sucesso é um dos pontos mais frustrantes do desenvolvimento de *software*, pois resulta em oportunidades perdidas e insatisfação de clientes. Assim sendo, além de outros fatores de qualidade, é importante que um produto de *software* seja desenvolvido com os recursos e cronograma previstos. O sucesso de um projeto depende muito da habilidade do gerente em estimar seus custos e prazos no início de seu desenvolvimento e controlá-los ao longo do processo de desenvolvimento (CRUZ, 1998).

[1] Dados quantitativos que caracterizam o ponto de partida de um projeto (JONES, 2000).

Quando o tema em pauta são os custos e prazos do projeto, a definição e utilização de bons processos para sua gerência é de grande importância. O objetivo de processos para gerência de custos e prazos é fornecer diretrizes que devem ser seguidas para a realização das estimativas de um projeto e, com seu desenvolvimento, direcionar as atividades de acompanhamento e controle, de forma a auxiliar na proximidade entre os valores estimados e os reais.

Sendo assim, a utilização de processos para gerência de custos e prazos bem definidos pode auxiliar na realização de estimativas com menor margem de erro, indicar caminhos de acompanhamento e controle e, dessa forma, tornar menos frequentes e menores os desvios dos projetos, favorecendo o sucesso de um maior número de projetos de *software*.

Porém, apenas concluir um projeto no prazo e orçamento previstos não mais indica ser este um projeto de sucesso. É necessário "agregar valor" à organização. Esse valor pode ser representado pelo conhecimento adquirido pela organização no desenvolvimento do projeto. Para trabalhar com a captura e utilização de conhecimento devem ser utilizados os conceitos e práticas da gerência do conhecimento.

A gerência do conhecimento tem sido reconhecida pelas organizações em geral como um importante fator de sucesso, uma vez que as constantes mudanças de mercado, tecnológicas e sociais exigem rápidas tomadas de decisão e atualizações em procedimentos, métodos e até estruturas organizacionais (O'LEARY e STUDER, 2001; ABECKER *et al.*, 2001). A gerência do conhecimento trata a descoberta, aquisição, criação, disseminação e utilização de conhecimento, contribuindo para o constante aprendizado organizacional.

O crescente interesse pela gerência do conhecimento chegou recentemente à indústria de *software* e alguns pesquisadores têm trabalhado com a aplicação de seus conceitos em organizações que desenvolvem e mantêm *software* (MARKKULA, 1999; VILLELA *et al.*, 2001a; MENDONÇA *et al.*, 2001; WANGENHEIM *et al.*, 2001; RUS e LINDVALL, 2002).

O conceito de Ambientes de Desenvolvimento de *Software* Orientados à Organização (ADSOrg), em cujo contexto este trabalho se insere, surgiu da necessidade de gerenciar o conhecimento organizacional adquirido ao longo de projetos de *software*. Esses ambientes apoiam a atividade de Engenharia de *Software* em uma organização, fornecendo

o conhecimento acumulado e relevante para essa atividade e dando apoio ao aprendizado organizacional em Engenharia de *Software* (VILLELA *et al.*, 2001a).

O conhecimento para gerência de custos e prazos de projetos de *software* é um exemplo de conhecimento presente em uma organização que desenvolve e/ou mantém *software*. Durante o planejamento de um projeto de *software* é realizado o planejamento de custos, no qual esforço, prazo e custos propriamente ditos são planejados para o projeto em questão. Quanto maior a experiência e conhecimento do gerente do projeto, maior será sua capacidade de realizar um bom planejamento de custos.

Ambientes de Desenvolvimento de *Software* Orientados à Organização se propõem a apoiar as atividades de Engenharia de S*oftware*, possibilitando a gerência do conhecimento que pode ser útil aos engenheiros de *software* ao longo dos projetos de uma organização (VILLELA *et al.*, 2000). Assim, o conhecimento de gerência de custos e tempo adquirido em projetos da organização é uma das áreas de conhecimento que deve ser gerenciada por esses ambientes. O trabalho aqui descrito está inserido nesse contexto.

1.2 Objetivo da Tese

O principal objetivo deste trabalho é estabelecer um meio de auxiliar a elaboração do Plano de Custos e cronograma do Projeto, baseando-se no uso de técnicas e práticas utilizadas na Engenharia de *Software* e no aprendizado da organização por meio de suas experiências passadas, através da gerência do conhecimento.

Para que esse objetivo seja alcançado, foram definidos processos de gerência de custos e gerência de tempo, tendo como base a gerência do conhecimento organizacional, que deve auxiliar a realização das estimativas de custos e prazos e facilitar o acompanhamento destes ao longo do processo de desenvolvimento, através da disponibilização do conhecimento acumulado em projetos anteriores, armazenado no Repositório de Conhecimento Organizacional de um Ambiente de Desenvolvimento de *Software* Orientado à Organização.

1.3 Organização da Tese

Esta dissertação é composta por cinco capítulos, além desta Introdução.

No Capítulo 2, *Planejamento de Projetos e Estimativas de Custos*, são apresentados estudos na área de planejamento de custos e resultados de comparações entre várias técnicas obtidos em pesquisas realizadas em projetos de empresas de várias partes do mundo. São também descritos dois modelos paramétricos para estimativas de custos: Análise de Pontos de Função e COCOMO II.

No Capítulo 3, *Ambientes de Desenvolvimento de Software Orientados à Organização e a Estação TABA*, são apresentadas as principais características de Ambientes de Desenvolvimento de *Software* Orientados à Organização, os conceitos de gerência do conhecimento e a estrutura da Estação TABA.

No Capítulo 4, *Planejamento de Custos em Ambientes de Desenvolvimento de Software Orientados à Organização* são apresentados os processos de gerência de custos e gerência de tempo propostos neste trabalho e a abordagem proposta para gerência de custos e gerência de tempo em Ambientes de Desenvolvimento de *Software* Orientados à Organização.

No Capítulo 5 é descrita a ferramenta *CustPlan*, desenvolvida na Estação TABA, para apoiar os processos de gerência de custos e gerência de tempo definidos no capítulo 4.

No Capítulo 6 são apresentadas as conclusões, contribuições da Tese e perspectivas de trabalhos futuros.

Capítulo 2

Planejamento de Projetos e Estimativas de Custos

Neste capítulo são apresentados alguns conceitos relacionados a Estimativas de Custos e relatos de pesquisas realizadas nesse contexto. Também são apresentadas duas técnicas de estimativas: Análise de Pontos de Função e COCOMO II.

2.1 Introdução

Atualmente, profissionais que ocupam cargos de gerência de projetos têm assumido um importante papel quando o assunto em foco são projetos de *software*.

Assim como os líderes na indústria de Tecnologia de Informação, os gerentes de projeto criam estratégias e orquestram cuidadosamente a elaboração de planos de ação que sejam capazes de gerenciar o desenvolvimento de projetos de maneira eficiente (MURCH, 2000).

A função de gerência pode ser conceituada como o conjunto de atividades e tarefas realizadas por uma ou mais pessoas, com o objetivo de planejar e controlar as atividades de outras para atingir um objetivo ou completar uma atividade que não pôde ser atingido ou completada de forma independente.

PRESSMAN (2001) afirma que a gerência de projetos de *software* envolve planejamento, acompanhamento e controle de pessoas, processos e eventos que ocorrem à medida em que o *software* evolui de um conceito inicial a uma implementação operacional.

Dentre as atividades da gerência determinadas por PRESSMAN (2001), destaque especial merece a de planejamento, que deve ser realizada antes da execução propriamente dita do projeto.

Para realizar um projeto é necessário que haja o planejamento de quais ações deverão ser tomadas, como elas deverão ser realizadas e qual o contexto em que elas estão inseridas. O planejamento do projeto engloba o planejamento do processo de desenvolvimento, recursos humanos, *hardware* e *software* necessários, custos relacionados, cronograma, qualidade do produto e do processo, documentação e riscos envolvidos.

Todo planejamento realizado deve ser registrado em um documento chamado Plano do Projeto. Cada item a ser analisado e planejado origina um plano específico, como por exemplo: Plano de Custos, Plano de Riscos, Plano de Recursos Humanos e Plano da Qualidade. O conjunto de todos os planos específicos que envolvem o projeto constitui o Plano do Projeto.

O Plano do Projeto é o elemento base para o processo de gerência do projeto, pois se comporta como o ponto inicial do controle, onde as informações básicas iniciais são registradas e, com o desenvolvimento do projeto, serão modificadas e executadas. Seu objetivo é prover um conjunto de diretrizes que habilite o gerente a realizar estimativas razoáveis e análises satisfatórias das variáveis pertinentes ao projeto, como por exemplo qualidade, riscos, prazos e custos, facilitando o desenvolvimento do projeto dentro das expectativas estabelecidas pelas partes envolvidas.

Embora existam várias abordagens para o processo de *software*, todas elas enfatizam a importância da realização do planejamento. DONALDSON e SIEGEL (2000), por exemplo, localizam o Plano do Projeto em uma visão "figurativa" da gerência de projetos de *software* através das fases do desenvolvimento de *software*, definindo-as de forma macroscópica como "o que", "como", "construção" e "uso". Baseado nessas macrofases, os autores propõem um modelo genérico para a Engenharia da Informação e um ciclo de vida genérico para desenvolvimento de *software*, onde se insere o Plano do Projeto. Para os autores, o planejamento deve ser realizado nas fases "o que" e "como" e o acompanhamento e controle seriam executados nas fases "construção" e "uso".

2.2 Estimativas de Custos

Quando os computadores começaram a ser utilizados em escala comercial os custos com *software* representavam menos de 20% dos custos totais com os sistemas de computação. Com o tempo, essa relação foi se alterando e os custos com *software* passaram a representar uma fatia muito mais expressiva dos custos totais. Esse crescimento impulsionou o interesse, por parte das organizações, pela medição e controle dos custos dos projetos de desenvolvimento de *software*.

Entregar um produto no prazo, dentro do orçamento e com um nível de qualidade, no mínimo, aceitável, tornou-se um fator crítico para muitas organizações. Subestimar os

custos de um projeto de *software* pode comprometer a qualidade do produto a ser entregue. Por outro lado, superestimá-los pode levar à perda de competitividade no mercado.

BROOKS (1975), na década de 70, já lançava a reflexão:

" - Como um projeto atrasa um ano?

- Um dia de cada vez..." .

Segundo ele, constatadamente, a maioria dos fracassos de projetos se dá devido a atrasos de cronogramas mal elaborados e estimativas mal realizadas.

Em resposta a essa questão, nas últimas duas décadas, muitos estudos têm sido realizados na área de estimativas de custos para projetos de *software*. As abordagens existentes para a realização das estimativas de projetos de *software* são: *(i)* modelos paramétricos; *(ii)* analogia de estimativas; e, *(iii)* julgamento de especialistas. Essas abordagens são descritas a seguir.

2.2.1 Modelos Paramétricos

Os modelos paramétricos utilizam características do projeto em modelos matemáticos e/ou algoritmos para calcular as estimativas do projeto. Alguns desses modelos, já considerados clássicos como o COCOMO II (evolução do COCOMO 81) (BOEHM *et al.*, 2000; TRINDADE *et al.*, 1999) e a Análise de Pontos de Função (GARMUS e HERRON, 2001) têm sua aplicabilidade constatada em experiências e práticas de muitas organizações, registradas em artigos e outras publicações[2].

Outras técnicas também têm sido estudadas e analisadas. RUNESON *et al.* (2000) avaliaram as técnicas MARK II PF, *Full Function Point* e *Bispoke*. JEFFERY *et al.* (2000) realizaram experiências com Estimativas por Analogias, Regressão OLS, *Stepwise*, ANOVA, CART e Regressão Robusta.

Analisando algumas técnicas existentes e considerando-as relativamente de difícil utilização, YAMAURA e KIKUNO (1999) propuseram o TCE – *Top-down Cost Estimation*, apresentando-o como rápido, fácil e capaz de gerar estimativas acuradas. O TCE é composto por três passos básicos que são realizados consultando-se tabelas fornecidas pela técnica e ajustadas para cada organização. Os passos básicos são: (1)

[2] Essas técnicas serão descritas na seção 2.3.

identificar a classificação funcional do *software* e verificar o custo padrão para este tipo de *software*; (2) ajustar o custo padrão considerando as estratégias de negócio do projeto, como por exemplo: a prioridade é manter a qualidade do projeto ou a prioridade é manter a data final; e, (3) reajustar o custo considerando características do ambiente de desenvolvimento.

HASTINGS e SAJEEV (2001) realizaram uma análise dos principais métodos de estimativas e medição e propuseram um novo método, baseado em vetores. Na verdade, são dois novos métodos: VSM – *Vector Size Measurement* e VPM – *Vector Prediction Model.* O primeiro tem como objetivo medir o tamanho do *software* e classificar os sistemas de *software*. O segundo, por sua vez, estima o esforço necessário ao desenvolvimento do projeto.

Os autores utilizaram as considerações de Fenton (FENTON, 1991) quando este afirma que o tamanho do *software* é função, entre outros atributos, de sua funcionalidade e complexidade. Fazendo uso dessa consideração, o VSM traça um vetor que representará a "magnitude" do sistema em função de um plano cartesiano funcionalidade x complexidade. Diversas fórmulas e relações foram propostas para obter o traçado do vetor e realizar sua interpretação para o projeto em análise.

O VPM tem como objetivo estimar o esforço provável para o desenvolvimento do projeto. Para tal, os requisitos do sistema devem estar representados em uma linguagem especial, chamada ASL (*Algebraic Specification Language*). Após essa especificação, podem ser utilizados métodos baseados em produtividade, regressão linear ou modelos de custos para que, baseando-se em suas fórmulas e relações, seja possível calcular o valor da estimativa do esforço.

MIRANDA (2001), por sua vez, realizou uma proposta para que seja possível obter uma noção mais exata do tamanho do projeto. Uma "entidade" do sistema é escolhida como referência e todas as demais serão mensuradas em função da original, através da utilização de uma matriz de julgamento. Esse autor ressalta, ainda, que o problema da realização *ad hoc* de estimativas não é acadêmico, pois as técnicas existem. O problema é prático, pois elas não são utilizadas, gerando orçamentos e cronogramas questionáveis.

Nos últimos vinte anos, muitos estudos foram realizados comparando-se as técnicas de estimativas de custos para *software*. Nos anos 80, modelos paramétricos foram

utilizados (BOEHM, 1981; PUTNAM, 1978) e comparados em conjuntos de dados de projetos de diversos tamanhos e ambientes. Algumas das principais conclusões foram que esses modelos geravam resultados fracos quando aplicados sem calibração em outros ambientes. Kemerer (BRIAND *et al.*, 1999), por exemplo, usou 15 projetos de aplicações para negócios e comparou quatro modelos: SLIM (BOEHM, 1981), COCOMO (BOEHM, 1981), *Estimacs* (PUTNAM, 1978) e Análise de Pontos de Função (GARMUS e HERRON, 2001). A margem de erro encontrada foi consideravelmente alta (85%), o que levou à reavaliação das técnicas testadas, surgimento de novas técnicas e desenvolvimento de procedimentos de calibração para os modelos paramétricos, permitindo que cada organização possa utilizar valores diferentes para as constantes e outros parâmetros dos modelos, adquirindo-os através da análise de dados e projetos da própria organização, o que diminuiu consideravelmente a margem de erro das estimativas realizadas pelos modelos paramétricos. O COCOMO II (BOEHM *et al.*, 2000) é um bom exemplo de modelo que permite calibração para cada organização em particular.

Nos anos 90, os estudos também envolveram modelos não paramétricos, baseados em algoritmos inteligentes e analogias, que são descritos a seguir.

2.2.2 Estimativas baseadas em Analogias

As estimativas baseadas em analogias são métodos não paramétricos que utilizam dados históricos de outros projetos para realizar as estimativas para o projeto corrente. As analogias são realizadas levando-se em consideração características comuns aos projetos. São frequentemente utilizadas nas estimativas de custos totais do projeto quando existe uma quantidade limitada de informações detalhadas sobre ele, como por exemplo nas fases iniciais. Também são utilizadas para apoiar a distribuição do tempo e custos totais do projeto em suas fases, módulos e/ou atividades (PMBOK, 2000).

Muitos estudos de caso têm sido desenvolvidos para analisar a utilização da analogia de estimativas e os resultados têm sido consideravelmente positivos.

SHEPPERD e SCHOFIELD (1997), por exemplo, realizaram comparações avaliando estimativas por analogias e *stepwise regression*. Seus resultados mostraram que estimativas por analogia superavam *stepwise regression*. Em 299 projetos de 17

organizações foi possível constatar que os modelos baseados em algoritmos inteligentes e analogias superavam os paramétricos, sendo o COCOMO e a Análise de Pontos de Função os que apresentaram melhores resultados, dentre os modelos paramétricos.

OHLSSON e WOHLIN (1999) analisaram a realização das estimativas de esforço utilizando analogias observando 26 projetos realizados entre 1996 e 1998 . O objetivo era produzir valores que pudessem ser utilizados pelos estudantes do Departamento de Sistemas de Comunicação da *Lund University* como referência para estimarem seus projetos. Os autores dividiram os projetos em oito partes, que foram utilizadas para realizar as estimativas. São elas: requisitos, testes, subtestes, processos, fluxos, saídas, entradas e sinais. Observando as partes, o esforço e o número de membros envolvidos nos projetos realizados em 1996 e 1997 os autores realizaram analogias, análises e regressões matemáticas e propuseram valores médios para o esforço para cada parte do projeto. Em seguida, utilizando outros modelos de estimativas, foram realizadas as estimativas para os projetos de 1998. Também foram realizadas estimativas para os projetos de 1998 utilizando-se os valores propostos pelos autores em 1997 considerando as partes do projeto. Essas estimativas foram, então, comparadas e, na maior parte dos casos, as estimativas realizadas considerando-se as partes do projetos e os valores propostos pelos autores através das analogias foram mais acuradas. Os autores constataram que nos casos em que essas estimativas não eram melhores, a margem de erro encontrada era consideravelmente alta, o que os fez concluir que um estudo mais profundo sobre as partes que foram utilizadas para dividir os projetos deveria ser realizado, bem como repetir a pesquisa considerando mais projetos.

MENDES e COUNSELL (2000) desenvolveram um estudo sobre a realização de estimativas de esforço utilizando analogias em projetos Web. Nesse estudo, os autores utilizaram a ferramenta *ANGEL (ANaloGy softwarE tooL)*, desenvolvida pela *Bournemouth University*, que é capaz de realizar analogias para um projeto comparando suas características com as características de projetos armazenados em uma base de dados. Para realizar as analogias, foram utilizados dois conjuntos de dados contendo dados empíricos de projetos de desenvolvimento de *software* para Web. As analogias foram realizadas considerando o número de arquivos HTML da aplicação, o número de arquivos HTML que seriam reutilizados de outras aplicações, o número de *links* da aplicação e o grau de

interação dos arquivos da aplicação. Os resultados obtidos nesse estudo mostraram que a analogia de estimativas fornecia estimativas próximas dos valores praticados para os projetos.

2.2.2.1 A Base de Dados Históricos

Observando os resultados dos estudos que realizaram analogias com dados de projetos anteriores, a necessidade de armazenar em uma base de dados históricos as informações a respeito de projetos que já foram concluídos tornou-se evidente. Os valores realmente praticados em cada projeto podem, então, ser utilizados como pontos de apoio para as estimativas dos novos projetos através das técnicas de analogias. Diversos autores registraram os resultados de seus estudos para verificar a necessidade da base de dados históricos.

MIRANDA (2001), por exemplo, apresenta os resultados dos estudos de Albert L. Lederer e Jayesh Prasad, que mostram que utilizar dados históricos e documentar o projeto são atividades capazes de produzir estimativas muito melhores do que a intuição permite.

ENGELKAMP *et al.* (2000) argumentam que algumas técnicas de estimativas são muito complexas, o que acaba exigindo um conhecimento muito apurado de seus usuários. Sua proposta é prover mecanismos e técnicas de estimativas de custo, esforço e prazo que possam ser mais acessíveis aos gerentes de projeto. Dessa forma, a necessidade de um banco de dados contendo dados das experiências anteriores em projetos é por ele vislumbrada.

Um projeto piloto na sd&m AG (*Software Design & Management*), uma empresa de desenvolvimento de aplicações técnicas e sistemas de informação da Alemanha, originou uma proposta completa da base de experiências contendo: como iniciar uma base de experiências, quais seus parâmetros (o que armazenar de cada projeto – identificadores, variáveis quantitativas, variáveis qualitativas e classificadores) e como avaliar os dados armazenados.

BASILI *et al.* (2001a) reforçam a efetividade da utilização de dados históricos, considerando primitivas as metodologias de desenvolvimento aplicadas e propondo como solução a utilização de produtos, processos e experiências da própria organização, por

meio da gerência do conhecimento, através da base de dados históricos de projetos. Como experiência, realizou um trabalho na multinacional Q-Labs, onde um sistema de gestão de experiências foi implementado com base no conceito da Fábrica de Experiências.

BASILI *et al.* (2001b) tratam no trabalho realizado na Q-Labs a idéia de uma generalização do conceito da Fábrica de Experiências para organizações que não são desenvolvedoras de *software,* refletindo sobre o momento em que a base de dados históricos ainda está vazia, ou seja, quando nenhuma experiência foi registrada. Enfatiza a necessidade de não apenas armazenar os dados na base, relatando a dificuldade de analisá-los e de reutilizá-los.

Outra experiência trata de um estudo realizado na ESA (*European Space Agency*) (BRIAND *et al.*,1999; BRIAND *et al.*, 2000), mostrando que uma organização possuindo uma base de dados históricos própria, com informações dos projetos por ela desenvolvidos, tem menos chance de cometer erros ao realizar as estimativas de custos, analisando essas informações, do que uma organização que utiliza uma base de dados históricos multiorganizacional, isto é, com dados de projetos desenvolvidos por diversas organizações. Ainda assim, as organizações que acessam uma base de dados multiorganizacional têm possibilidade de cometer uma margem menor de erros do que aquelas que não realizam qualquer tipo de registro dos valores praticados em seus projetos.

A funcionalidade de uma base de dados própria para cada organização também é destacada por MOSES *et al.* (2000) quando os autores realizam estudos sobre como refinar as estimativas de esforço em pequenas empresas de desenvolvimento de *software*. Eles utilizam uma abordagem *Bayesiana* e observam as diferenças entre pequenas e grandes empresas, destacando como os dados da grande empresa seriam inúteis para estimar projetos na pequena empresa, devido às particularidades do domínio dos problemas e soluções.

ANGELIS *et al.* (2001), por outro lado, defendem que, em alguns casos, a utilização de uma base de dados multiorganizacional pode ser mais expressiva que uma base própria. Por exemplo, quando uma organização de *software* inicia suas atividades, ela ainda não tem experiências próprias para reutilizar, então, utilizar dados de outras empresas pode ser útil. Quando uma empresa muda o domínio de suas aplicações ou insere uma nova tecnologia, recorrer a uma base multiorganizacional pode auxiliá-la nas

estimativas do novo projeto. Os autores realizaram um estudo considerando uma base com dados coletados pelo *International Software Benchmarking Standards Group*. Essa base continha dados de 789 projetos categorizados de acordo com sua natureza (área de negócios, domínio organizacional, uso de certas ferramentas e métodos, dentre outras). Os autores propuseram um modelo estatístico para realização de estimativas de projetos, que foi proposto baseando-se em analogias realizadas na base de dados da pesquisa. Os resultados obtidos mostraram que o modelo proposto é eficiente, porém os autores ressaltam que o estudo deve ser repetido e que trabalhar com dados categorizados exige que muitos estudos sejam realizados antes de considerar o modelo realmente eficiente. Tomando como referência o universo considerado pelo estudo, a utilização da base de dados multiorganizacional mostrou-se eficiente.

Dada a importância da base de dados históricos é preciso considerar que em um determinado momento ela estará vazia. Como proceder para inserir os primeiros dados nela? E, de posse dos dados dos projetos da base de dados históricos, como agrupá-los? Como identificá-los como similares para que possam ser base de apoio para as estimativas de futuros projetos? É necessário traçar uma caracterização para os projetos. Essa caracterização deve ser capaz de responder às questões básicas para a definição dos perfis de projetos e, dessa forma, agrupá-los como similares.

Alguns autores têm registrado pesquisas e resultados que auxiliam a responder esses questionamentos.

Em resposta ao questionamento de armazenamento inicial de dados históricos, BOEHM (2000) apresenta diversas razões para a utilização do COCOMO II. A principal delas é a consideração de que a maioria dos métodos de estimativas só são realmente eficientes quando utilizados com base em dados de projetos anteriores (métodos não paramétricos). Esse fato faz com que Boehm considere esses métodos “ineficientes”, pois não são capazes de prover meios para estimar o primeiro projeto a ser armazenado na base de dados históricos, o que o COCOMO II faz, sem restrições. Essa capacidade atribuída ao COCOMO II foi obtida devido a seus fatores de calibração já serem fornecidos com base no estudo de dezenas de projetos com os mais diferentes perfis.

Uma outra forma de iniciar o armazenamento dos dados na base de dados históricos é utilizar o julgamento de especialistas, que será descrito na próxima seção (2.2.3).

Após os dados estarem armazenados na base, deve-se estabelecer critérios de caracterização para que os dados possam ser acessados e utilizados na analogia de estimativas.

Segundo IDRI e ABRAN (2001), o quesito similaridade entre projetos de *software* não tem sido assunto de estudos profundos mesmo sendo frequentemente utilizado em estimativas de esforço para o desenvolvimento de *software* baseadas na analogia entre projetos. Entretanto, a complexidade dos módulos e experiência dos programadores são citados como fatores que descrevem projetos de *software*. Além disso, é relatada a metodologia do COCOMO e COCOMO II que utilizam 17 atributos para descrever um projeto. Tais atributos referenciam questões de tamanho do *software*, forma de desenvolvimento, características do produto, da equipe, de tecnologia e restrições em geral. Os mesmos poderiam, então, ser considerados como pontos importantes para caracterizar os projetos e, posteriormente, agrupá-los como similares.

JONES (2000) considera que há sete tipos básicos de informação para traçar caracterização de projetos que são comuns entre avaliações, *benchmarks* e estudos de *baseline*, sendo elas: o país para o qual o projeto se destina, a cidade, a indústria, a natureza do projeto (novo desenvolvimento, melhoria (adição de funções), adaptação para atender a novas regulamentações, reparo de defeitos, melhoria de desempenho, migração para nova plataforma, nacionalização, reengenharia, atualização em massa (por exemplo: Euro e Ano 2000) ou híbrido), o escopo do projeto (subrotina, módulo, protótipo descartável ou protótipo evolutivo), a classe do projeto (aplicação para uso corporativo desenvolvida com recursos internos, aplicação para uso corporativo desenvolvida com a contratação de terceiros, aplicação para uso corporativo que atende a padrões militares, aplicação *shareware* ou *freeware*, aplicação para uso externo via Internet, aplicação para ser distribuída junto com dispositivos físicos, aplicação comercial para venda ou *leasing*, aplicação desenvolvida sob contrato de encomenda para terceiros, aplicação desenvolvida sob contrato de encomenda para o governo ou aplicação desenvolvida sob contrato de encomenda para instituição militar) e o tipo do projeto (não procedural, *web applet*, aplicação *batch*, ou aplicação interativa)

MENZIES e SINSEL (2000), por sua vez, retratam a necessidade de considerar a precedência, flexibilidade de desenvolvimento, análise arquitetural ou resolução dos riscos,

coesão da equipe de desenvolvimento, maturidade do processo, atributos do produto (confiabilidade requerida, tamanho da base de dados, complexidade do produto, nível de reuso, requisitos de documentação), atributos da plataforma (restrição de tempo de execução, armazenamento da memória principal, volatilidade da plataforma), atributos da equipe (capacidade do analista e do programador, continuidade do programador, experiência do analista, experiência com a plataforma), atributos do projeto (experiência com a linguagem e ferramentas, uso de ferramentas de *software*, desenvolvimento distribuído), confiabilidade do sistema e pressões de cronograma.

KURTZ (2001) alega que além das características sugeridas pelos demais autores, é de grande importância considerar características de investimento e da organização que atua no desenvolvimento do projeto.

Estabelecidos os critérios de caracterização dos projetos é preciso definir o mecanismo que indicará quais projetos são similares. SCHOFIELD e SHEPPERD (1995) usam a similaridade para estimar o esforço de desenvolvimento de *software* baseado em analogia e propõem uma abordagem na qual os projetos similares são encontrados medindo-se a distância Euclidiana em um espaço n-dimensional onde cada dimensão corresponde a uma variável (um critério de caracterização). A limitação desta abordagem é que ela não trabalha com variáveis medidas em escalas nominais (exemplo: domínio de aplicação). MENDES e COUSELL (2000) utilizaram em seus estudos a ferramenta *ANGEL (ANaloGy softwarE tooL)*, desenvolvida pela *Bournemouth University* e já citada anteriormente, que também utiliza distância Euclidiana para verificar a analogia entre os projetos.

IDRI e ABRAN (2001), por sua vez, propõem utilizar um modelo baseado em lógica *Fuzzy* para definir as métricas de similaridade dos projetos de *software* e então superar a limitação de não poder trabalhar com atributos de projeto medidos em escalas nominais .

PRIETO-DIAZ (1991) propôs a *classificação facetada*, buscando tornar possível a recuperação de componentes de *software* baseada em similaridade. Segundo o autor, selecionar componentes similares é um problema de classificação e o esquema de classificação é crucial no processo de reutilização de componentes. Seu esquema pode ser

adaptado de forma a tornar possível a recuperação de projetos de *software* baseada em similaridade.

Outra questão relativa à base de dados históricos que deve ser considerada é que, algumas vezes, os dados presentes podem estar incompletos. Isso pode ocorrer devido ao não registro dos dados no momento de alimentar a base de dados ou por problemas relacionados à integridade, procedimentos e segurança da base de dados. A ausência de alguns dados pode comprometer as estimativas realizadas por analogias. Para tratar casos de dados incompletos foram desenvolvidas *Missing Data Techiniques* (MDT) (FORD, 1976; RAYMOND e ROBERTS, 1987; KROMREY e HINES, 1994). STRIKE *et al.* (2001) observaram os problemas gerados pela perda de dados para o processo de estimativas de custos e realizaram um estudo comparando algumas MDT aplicadas a modelos de estimativas de custos baseados em dados históricos. Para realizar o estudo, os autores utilizaram uma base de dados composta por 206 projetos de 26 empresas finlandesas de diversas áreas (negócios, aplicações bancárias, manufatura e outras). Os projetos tinham seu tamanho medido em pontos de função, o esforço total fornecido em pessoas/mês e valores para quinze fatores de produtividade. Utilizando três mecanismos para eliminar alguns dados da base de dados (MCAR, MAR e NM (LITTLE e RUBIN, 1987)) os autores testaram dez diferentes técnicas de tratamento de dados perdidos. Foram realizadas 825 simulações, variando o número de variáveis perdidas de um a três e o percentual de dados perdidos de 5 a 40%. Os resultados obtidos mostraram que é possível trabalhar em modelos de estimativas com dados perdidos com uma margem de erro aceitável e até insignificante quando há poucos dados perdidos. À medida que a quantidade de dados e sua utilidade para as estimativas aumenta, também aumentará a margem de erro da estimativa realizada.

Uma última questão que ainda diz respeito à base de dados históricos utilizada na analogia de estimativas é quanto tempo os dados devem permanecer na base, ou seja, quando os dados dos projetos concluídos podem estar obsoletos para serem utilizados nas analogias e, por este motivo, devem ser excluídos ou desconsiderados. MENDES e COUSELL (2000) realizam este questionamento e afirmam que muitos estudos e observações ainda devem ser realizados, mas que a introdução de novas tecnologias e expressivas mudanças organizacionais ou até mesmo econômicas são pontos que devem ser

observados para a eliminação de dados de projetos da base utilizada na analogia de estimativas.

Os bons resultados obtidos nos estudos de analogia de estimativas não vêm desacreditar os modelos paramétricos. Alguns dos estudos desenvolvidos associaram métodos paramétricos e analogias, obtendo resultados que levaram à conclusão de que uma associação desses dois tipos de modelos seria capaz de fornecer estimativas com uma margem de erro aceitável, ou seja, em torno de 20%.

2.2.3 Julgamento de Especialistas

Quando a analogia de estimativas foi apresentada na seção anterior, argumentou-se sobre a necessidade de possuir uma base de dados históricos, definir os critérios de caracterização dos projetos e recuperar os dados de projetos concluídos que possam ser utilizados como base para as estimativas de novos projetos. Porém, também é preciso considerar que nem sempre a organização tem acesso a uma base de dados históricos de projetos. Algumas vezes ela não possui dados formais nos quais possa se basear para realizar as estimativas de um projeto nem experiência ou conhecimento para utilizar de forma eficaz algum modelo paramétrico de estimativas.

BOEHM e FAIRLEY (2000), ao analisarem essa situação, consideram uma outra forma de realizar as estimativas de um projeto. Eles afirmam que esta ainda é a forma mais comumente utilizada, chamada de opinião de especialistas, julgamento de especialistas ou utilização de experiência pessoal, e consiste no ato dos gerentes de projetos estimarem os valores para os projetos, baseando-se em suas próprias experiências passadas. A utilização da experiência pessoal, segundo argumentos dos autores, não é capaz de produzir dados históricos formais e, tipicamente, não apresenta regras para sua abordagem, além de não permitir calibrações para melhorar as estimativas mal realizadas, uma vez que não há padrão para sua realização.

Os autores ressaltam, ainda, que as técnicas de estimativas baseadas em modelos matemáticos ou em analogias podem ser utilizadas para uma diversidade considerável de tipos de projetos, pois são capazes de considerar as diferenças e especialidades de cada projeto em particular. Para um especialista é consideravelmente difícil ter experiência em

todos os tipos de projetos de desenvolvimento de *software*, o que pode levá-lo a estimar de forma ineficiente um projeto com características diferentes das que trabalhou até o presente momento.

Para BOEHM e FAIRLEY (2000) um especialista é capaz de realizar estimativas que representem desvios aceitáveis quando comparadas aos valores realmente praticados se ele tiver repetido a atividade de estimar muitas vezes e, com isso, tiver alcançado para si um certo grau de maturidade para realizar as estimativas de um projeto.

Apesar de possuir algumas desvantagens, muitas vezes a utilização da experiência pessoal pode ser o caminho disponível para uma organização realizar as estimativas de seu primeiro projeto e iniciar, assim, a alimentação de uma base de dados históricos que possa ser utilizada nos projetos subsequentes.

BOEHM e FAIRLEY (2000) consideram, ainda, uma outra situação onde a opinião de um especialista é necessária e tem o caráter de julgamento e decisão. Nela, o gerente do projeto utiliza sua experiência para decidir os valores das estimativas quando técnicas diferentes apresentam valores diferentes para as mesmas variáveis do projeto (prazo, custos e esforço) ou para ajustar os valores propostos pelas técnicas. Segundo os autores, nesse caso, a experiência do especialista representa um fator crítico para a decisão dos valores das estimativas que se aproximem o máximo possível dos valores realmente praticados. MENDES e COUNSELL (2000) também consideram esse caso como o melhor caminho para a utilização da opinião de especialistas.

Apesar das desvantagens apresentadas pelos autores acima citados, alguns estudos têm sido realizados para mostrar que a opinião e julgamento de especialistas é um bom caminho para realização de estimativas. GRAY *et al.* (1999) realizaram estudos para analisar a realização de estimativas através da opinião de especialistas. Em um estudo de caso comparativo entre Análise de Pontos de Função, COCOMO e julgamento de especialistas os autores puderam observar que as estimativas realizadas pelos especialistas superavam as demais em acurácia e consistência. Neste estudo de caso os autores também puderam perceber que a maioria das empresas envolvidas no estudo utilizavam o julgamento de especialistas, mesmo quando utilizavam outros modelos. Ao fim do estudo, concluíram que uma associação formal das estimativas por analogias com o julgamento de especialistas é capaz de produzir estimativas satisfatórias, pois o julgamento de

especialistas teria a função de calibrar os valores apresentados pelos projetos similares. Os autores ressaltam que, mesmo com os resultados positivos, os riscos desse método não podem deixar de ser considerados.

Um outro estudo realizado por JOHNSON *et al.* (2000) também foi útil para analisar a acurácia das estimativas geradas por especialistas. Eles desenvolveram um pacote de ferramentas de estimativas denominado LEAP (*Lightweight, Empirical, Antimeasurement dysfunction and Portable)* e o utilizaram em um estudo de caso na *University of Hawaii* que envolveu 16 estudantes. A LEAP permite a realização das estimativas por modelos algorítmicos e/ou considerando a opinião do gerente do projeto. Considerando 128 projetos (cada estudante participou de oito projetos), os autores puderam perceber que as estimativas realizadas pela opinião dos estudantes tornavam-se melhores a cada projeto e que, na maioria dos casos, eram melhores que as calculadas pelos modelos algorítmicos. Inicialmente, foram considerados os três primeiros projetos de cada estudante, para alimentar uma base de dados históricos de projetos. Em seguida, outros projetos foram realizados e os estudantes passaram a observar as estimativas dos projetos anteriores e compará-las com os valores praticados, percebendo, assim, o erro cometido. A melhoria nos valores estimados nos cinco projetos subsequentes de cada aluno foi expressiva. A média de erro das estimativas de tamanho realizadas pelos estudantes no terceiro projeto era de 50% e diminuiu para 15% no oitavo. Para esforço, o erro diminuiu de 25% para 10%.

Os autores HOST e WHOHLIN (1997) realizaram estudos sobre a utilização do julgamento de especialistas como método de estimativas, analisando-o em um pequeno experimento baseado no PSP – *Personal Software Process* (HUMPREY, 1995) e constataram que as estimativas realizadas eram eficientes quando comparadas com os valores reais dos projetos.

Estudos recentes mostram que utilizar os três tipos de modelos de estimativas é um bom caminho para obter estimativas acuradas, uma vez que os pontos fracos de um modelo podem ser minimizados pelos outros modelos.

Como exemplo de utilização de técnicas diferentes para realizar as estimativas de um projeto, temos BIELAK (2000), que realizou um projeto para os geocientistas do Departamento de Serviços Técnicos e Exploração da Pesquisa da empresa Arco, atuante no

segmento petrolífero, onde pôde analisar a utilização da experiência pessoal de gerentes para realizar as estimativas de tamanho de um projeto e a utilização de dados históricos para melhorar essas estimativas. Para estimar os custos, prazos e esforço necessários para desenvolver o projeto foi utilizado o COCOMO II, que necessita, entre outros fatores, do tamanho do sistema para que seja possível calcular as demais estimativas. Ninguém na equipe do projeto era treinado em Análise de Pontos de Função, então o tamanho inicial do sistema precisou ser estimado considerando-se a experiência do gerente em projetos anteriores. Estimados os valores iniciais para uma parte do projeto, esta era desenvolvida. Concluído seu desenvolvimento, o tamanho real era, então, comparado com o tamanho estimado inicialmente. O processo se repetiu até que todo o projeto estivesse concluído e, a cada estimativa de tamanho realizada, os dados das partes já concluídas do projeto eram consultados. O resultado do estudo realizado pelo autor nesse projeto mostrou que, à medida que o projeto era desenvolvido e que mais dados históricos eram obtidos, menor era o desvio entre os tamanhos estimado e real do projeto.

2.2.4 Outras Questões sobre a Realização de Estimativas

Mesmo existindo diversas técnicas para a realização das estimativas de um projeto e estas apresentarem diversas abordagens diferentes (analogias, modelos matemáticos e experiência pessoal) uma dificuldade comum apresentada pelos gerentes de projeto está em realizar as estimativas no início do projeto, quando pouco é conhecido do mesmo. As estimativas iniciais são realizadas quando o levantamento dos requisitos ainda não foi completamente executado.

CHRISTENSEN e THAYER (2001) afirmam que realizar as estimativas na fase inicial do projeto é uma tarefa árdua e que traz consigo riscos, como por exemplo a baixa precisão dos requisitos que ainda não estão formalizados. Dados de projetos similares podem ajudar (PMBOK, 2000), mas pode ocorrer a não existência de projetos similares ou uma base de dados históricos que ainda esteja vazia. Uma forma de amenizar o problema das estimativas iniciais é realizar essas estimativas considerando apenas as grandes fases do projeto. Quando as informações do projeto tornarem-se mais precisas, o gerente pode realizar novas estimativas, agora com informações mais completas e peculiares ao projeto,

considerando todas as atividades que devem ser executadas ao longo de seu desenvolvimento.

RAINER e SHEPPERD (1999) realizaram um estudo durante vinte meses em um projeto na IBM para analisar o comportamento do cronograma do projeto ao longo de seu desenvolvimento. Inicialmente, o gerente do projeto propôs um cronograma considerando apenas as grandes fases do projeto. À medida em que o projeto era desenvolvido reuniões entre membros do projeto e usuários chave eram realizadas. Durante essas reuniões o cronograma era analisado e, sempre que necessário, era revisto e tinha suas estimativas de prazo e esforço refinadas. No total foram realizadas sete revisões do cronograma. O gerente do projeto considerou o projeto sendo de sucesso, pois além de ter satisfeito as exigências do cliente, foi entregue no limite de prazo e custos previstos e apresentou um ganho de mercado para a empresa. Como resultado do estudo, os autores enfatizaram a necessidade do replanejamento durante o desenvolvimento do projeto, afirmando que, apesar do esforço para realizar o planejamento nas fases iniciais do projeto ser intenso, o planejamento deve continuar fase após fase do projeto. Então, à medida que mais detalhes sobre o projeto são disponibilizados, as mudanças e refinamento nas estimativas devem ser realizados.

Considerando todas as pesquisas e práticas desenvolvidas na área de estimativas de projetos de *software*, ARMOUR (2002) realiza considerações sobre algumas questões que são consideradas mitos no processo de realização de estimativas em projetos de *software*. Ao retratar cada mito, o autor expõe o contexto em que o mesmo está inserido e o justifica. Algumas conclusões apresentadas por ARMOUR (2002) são: *(i)* é muito difícil produzir estimativas acuradas, pois os dados necessários para a realização destas só estarão disponíveis nas fases conclusivas do projeto; *(ii)* o tamanho do projeto não é capaz de, sozinho, determinar as estimativas. Outras variáveis como reuso e conhecimento necessário também devem ser consideradas; *(iii)* a utilização de dados históricos é útil, mas também não é capaz de garantir a acurácia das estimativas, pois os critérios de similaridade devem ser bem estabelecidos e os dados disponibilizados devem ser analisados e bem interpretados para serem aplicados ao projeto corrente; *(iv)* um coeficiente médio de produtividade não indica a produtividade real de cada indivíduo, sendo assim, estimativas geradas a partir desse coeficiente podem não ser precisas; *(v)* linhas de código e outros sistemas de medida,

como Análise de Pontos de Função, podem não refletir o tamanho real do projeto, pois podem não considerar reutilização, linguagens e aplicativos de desenvolvimento com facilitadores, como geradores de código, por exemplo; *(vi)* aumentar o número de pessoas em um projeto não significa que o mesmo poderá ser concluído em menos tempo, pois as tarefas podem ser dependentes. Finalizando, o autor ressalta que o processo de realizar estimativas é um processo de previsão e não de precisão, sendo assim, uma margem de erro deve ser determinada e as estimativas que estiverem um desvio menor ou igual a essa margem de erro devem ser consideradas acuradas.

2.3 Descrição de Algumas Técnicas de Estimativas de Custos

Como mencionado anteriormente, muitas pesquisas têm sido realizadas com experiências para observar o comportamento das diversas técnicas de estimativas e para tentar traçar um caminho para a realização das estimativas utilizando as melhores técnicas para cada projeto em particular. Neste trabalho, dar-se-á um maior foco à analogia de estimativas, através do uso do conhecimento (a ser descrito posteriormente) e experiências passadas, e a duas técnicas paramétricas: Análise de Pontos de Função e COCOMO II. Uma visão geral dessas técnicas paramétricas e estudos envolvidos são apresentados a seguir. A descrição dos passos para sua utilização, bem como exemplos de aplicação das técnicas, são realizadas nos Anexo 1 e Anexo 2.

2.3.1 A Técnica Análise de Pontos de Função

A Análise de Pontos de Função (APF) é um sistema de medidas proposto por Allan Albrecht, da IBM, em 1979, que quantifica as funções contidas em um *software* em termos que são significativos para seus usuários e fornece o tamanho do *software* em número de pontos de função.

Segundo JONES (1999), durante muitos anos, a indústria de *software* considerou estimar o tamanho de um sistema como sendo um problema difícil e, até mesmo, sem solução. Porém, nos últimos 20 anos, muitas técnicas têm sido propostas e a Análise de

Pontos de Função tem se destacado por ser utilizada em uma quantidade considerável de empresas e institutos no mundo todo.

Essa técnica recebe o apoio do IFPUG *(Institute Function Point Users Group)*, que é responsável por determinar e documentar as regras de aplicação e valores utilizados na Análise de Pontos de Função, bem como fornecer apoio e treinamentos aos usuários da técnica. Esse órgão possui comitês de representação em diversos países, onde é possível certificar indivíduos na utilização e conhecimento da técnica.

Em 1994, o IFPUG reconheceu a importância do apoio da ISO (*International Organization for Standardization)* para que a APF tivesse uma utilização mais expressiva no mercado. Sendo assim, a ISO reconheceu a APF como um padrão internacional. Estudos foram realizados para a elaboração de uma norma que contemplasse a abordagem da APF e chegou-se à conclusão de que a utilização de uma técnica específica poderia restringir a aplicação da norma. A comunidade ISO, então, baseada nos princípios da APF, publicou a ISO/IEC 14143 – *Information Technology – Software Measurement – Functional Size Measurement*, onde faz-se referência aos métodos FSM *(Functional Size Measurement)*, sem exigir a aplicação de um método específico, mas sendo a APF um dos métodos disponíveis. FSM refere-se a uma classe geral de técnicas de medição de *software* que considera os requisitos funcionais identificados pelo usuário como a base para calcular o tamanho do *software* (DEKKERS, 1999). A ISO/IEC 14143-1 define os conceitos fundamentais da FSM e descreve princípios gerais para aplicar um método FSM.

A Análise de Pontos de Função reflete a funcionalidade fornecida pelo *software* ao negócio em que está inserido. Dessa forma, pode ser aplicada em vários contextos de desenvolvimento, da fase inicial de especificação dos requisitos do sistema, até em momentos posteriores à sua implantação. Assim, indicadores como a produtividade do processo de desenvolvimento e o custo por ponto de função podem ser obtidos.

Cada uma das funcionalidades necessárias ao sistema recebe um peso numérico em função de seu tipo e sua complexidade. Esses pesos são totalizados em uma medida inicial de tamanho, que será normalizada através da ponderação de características gerais do sistema e fornecerá um resultado final que mede o tamanho e a complexidade do mesmo (GARMUS e HERRON, 2001).

Como mencionado anteriormente, Análise de Pontos de Função pode ser utilizada em diversos momentos do desenvolvimento de *software*, podendo medir um projeto que será desenvolvido, as manutenções em um sistema existente ou uma aplicação instalada. A contagem em pontos de função é realizada através da medida de dois tipos de funções que estão presentes em um sistema: *Funções Tipo Dados* e *Funções Tipo Transação*. As Funções Tipo Dados representam as funcionalidades fornecidas pelo sistema ao usuário, para atender às necessidades referentes aos dados que o sistema irá manipular. Trata-se dos arquivos internos e externos que são utilizados pelo sistema. As Funções Tipo Transação representam as funcionalidades de processamento dos dados fornecidas pelo sistema ao usuário. Trata-se das entradas, saídas e consultas do sistema.

Considerando que sistemas com complexidade e características diferentes podem apresentar um mesmo número de pontos de função (ou números muito próximos), a técnica propõe a avaliação de quatorze características que indicam o perfil do projeto. A cada característica, o usuário indica um nível de influência que pode variar de zero a cinco. Os valores atribuídos às características serão utilizados para ajustar o número de pontos de função encontrado inicialmente, refletindo, assim, as particularidades do projeto.

O número de pontos de função é uma medida de tamanho para o sistema e, a partir dela, podem ser obtidas relações que auxiliam na realização das estimativas dos projetos. Para estimar os custos de um projeto, relações como o esforço necessário para realizar um ponto de função e/ou o custo de um ponto de função são imprescindíveis. Esses valores podem ser obtidos através de consultas a tabelas fornecidas pelo IFPUG ou através da análise dos dados de projetos anteriormente realizados pela organização ou por outras.

JONES (1999) realizou um estudo observando dados de diversos projetos. Os resultados obtidos foram relações de um ponto de função com o número de documentos gerados, com a quantidade de código, com número de erros e com a quantidade de casos de testes.

O objetivo de JONES (1999) ter analisado a quantidade de documentos gerados para um ponto de função foi o tempo e esforço consumidos para gerar os diversos tipos de documentos que são necessários durante o desenvolvimento de um *software*. O autor chegou à conclusão que, dependendo do domínio da aplicação, cerca de 50% do custo total do *software* é atribuído à elaboração dos documentos do projeto. O autor dividiu os

documentos dos projetos em seis tipos: requisitos do usuário, especificações funcionais, especificações lógicas, planos de testes, manuais do usuário e documentos de referência para o usuário. Foram analisados projetos de quatro domínios diferentes. Para cada domínio foi calculado o intervalo médio da quantidade de cada tipo de documento produzido em um ponto de função. A utilização da relação entre o número de documentos gerados e um ponto de função já está presente em algumas ferramentas comerciais de estimativas de projetos.

Um outro ponto analisado por JONES (1999) foi a quantidade de código gerada por um ponto de função. O autor considerou diversas linguagens. Essa relação é bastante conhecida desde o início da utilização da APF.

JONES (1999) também analisou o número de erros e defeitos observados em um ponto de função e o número de casos de testes necessários para realizar um ponto de função. Para analisar os erros considerou as fases: requisitos, projeto, implementação, documentação, manutenção. Para cada fase o autor calculou o intervalo médio do número de erros encontrados em um ponto de função. Para analisar os casos de testes considerou: testes de unidade, testes de novas funções, testes de regressão, testes de integração e testes do sistema. Para cada tipo de teste o autor calculou o intervalo médio do número de casos de testes necessários para um ponto de função.

Com a utilização do paradigma de desenvolvimento orientado a objetos alguns autores desenvolveram estudos propondo adaptações da técnica APF para considerar os diagramas gerados na fase de análise de requisitos de projetos orientados a objeto. Alguns autores chegaram a propor ferramentas que realizam a contagem sem que seja necessária a interferência humana. Como exemplo desses autores temos KUSUMOTO *et al.* (2000) e UEMURA *et al.* (1999) que realizaram estudos envolvendo a aplicação da APF e adaptações em especificações de requisitos orientadas a objeto. Um estudo de caso foi realizado no sistema REQUARIO da empresa *Hitachi Ltd*. no qual foi utilizado o paradigma de orientação a objeto e especificação em UML *(Unified Modeling Language)*. Os autores desenvolveram uma adequação da APF que é capaz de analisar os diagramas em UML e fornecer o número de funções tipo dados, funções tipo transação e pontos de função do sistema. Essa adequação foi implementada em uma ferramenta que realizou as estimativas do sistema em estudo analisando os diagramas da especificação de requisitos.

Os resultados gerados pela ferramenta foram considerados adequados quando comparados com os resultados obtidos por um especialista em APF que realizou a medição manualmente. Os autores ressaltam que muitas aplicações da ferramenta em projetos ainda devem ser realizadas, pois o universo de testes foi relativamente pequeno, mas consideraram que a experiência representa um grande passo na automatização completa da APF. Um dos inconvenientes da utilização de uma ferramenta como essa para realizar as estimativas, é que, para isso, uma primeira especificação de requisitos do sistema deve estar pronta e representada em UML. Se a medição não for completamente automática, a especificação não precisa estar representada em tantos detalhes quanto em alguns dos diagramas. Uma das vantagens da automatização completa da técnica é evitar que pessoas diferentes obtenham estimativas diferentes para um mesmo projeto utilizando a técnica APF. Segundo os autores, isso ocorre pois pessoas diferentes têm visões diferentes de um mesmo sistema. Essa discrepância, no entretanto, não é desejada e pode ser resolvida com a utilização de uma ferramenta que realize as estimativas sem a interferência humana.

Assim, como apresentado no exemplo acima, a utilização do paradigma orientado a objetos gerou a necessidade de criar outros métodos ou adaptações da técnica APF para realizar as estimativas utilizando como base as especificações geradas nessa abordagem. Um dos métodos criados com esse objetivo é a contagem em *Pontos de Casos de Uso* (NAGESWARAN, 2001) .

Os diagramas de caso de uso são comumente utilizados para descrever a funcionalidades do sistema de acordo com sua forma de utilização pelos usuários. A técnica de análise de dimensão por casos de uso foi criada para permitir realizar estimativas para projetos orientados a objeto ainda na fase de levantamento de requisitos, utilizando-se os próprios documentos gerados nessa fase como subsídio para o cálculo das estimativas.

A técnica de estimativas por casos de uso foi proposta em 1993 por Gustav Karner, da *Objectory* (hoje, *Rational Software*). Ela baseia-se em dois métodos paramétricos: a Análise de Pontos de Função e MARK II, uma adaptação de APF, bastante utilizada na Inglaterra. A forma de dimensionar o sistema é a principal diferença dessa técnica com a Análise de Pontos de Função. Nela, considera-se o modo como os usuários utilizarão o

sistema, a complexidade das ações requeridas para cada tipo de usuário e uma análise em alto nível das operações do sistema. O nível de abstração é maior do que o da APF.

Uma vez que os casos de uso principais do sistema são levantados, é possível estimar o tamanho do sistema. O grau de detalhe dos casos de uso influencia diretamente no resultado final da medição. O ideal é medir apenas os casos de uso em nível de sistema, onde seja possível diferenciar transações e interações com o usuário. Assim, uma descrição de casos de uso em um nível muito alto (nível de negócio) ou muito baixo (casos de uso atômicos) não demonstra-se adequada para a medição.

Para realizar as estimativas utilizando a técnica Pontos de Caso de Uso são realizadas contagens dos atores e casos de uso do sistema obtendo uma contagem inicial. Em seguida, assim como na APF, características do sistema são analisadas para ajustar a contagem inicial e determinar o tamanho do sistema considerando suas principais características. A descrição detalhada dos passos para a realização de estimativas utilizando-se Pontos de Caso de Uso é realizada no Anexo 1.

2.3.2 O Modelo COCOMO II

COCOMO (*Constructive Cost Model*) foi proposto por Barry Boehm durante os anos 70 (BOEHM, 1981) e foi chamado de COCOMO 81. Após duas décadas, uma evolução, COCOMO II, foi desenvolvida no Centro de Pesquisa em Engenharia de *Software* da *University of Southen California* (USC).

COCOMO II é um modelo para a mensuração de esforço, prazos e custos. É um reflexo do estado de amadurecimento das tecnologias e da Engenharia de *Software* e traz adequações às mudanças ocorridas no decorrer das duas décadas, desde que o COCOMO original foi proposto (TRINDADE *et al.*, 1999).

Assim como seu predecessor, o COCOMO II realiza as estimativas de esforço, prazos e custos para o desenvolvimento de um *software* a partir da dimensão do mesmo. As estimativas de prazo e esforço são calculadas diretamente através da utilização das fórmulas propostas pelo modelo e do tamanho do sistema. A estimativa dos custos do projeto, por sua vez, pode ser obtida analisando-se os valores de prazo e esforço encontrados.

Para determinar as relações e os valores das variáveis do COCOMO II foi realizada uma pesquisa que utilizou dados de 161 projetos. Esses dados foram analisados e, através de regressões matemáticas, as relações e constantes foram definidas pelos pesquisadores.

O COCOMO II funciona como uma hierarquia de modelos que podem ser aplicados de acordo com a evolução do desenvolvimento do *software*. O primeiro deles, chamado de ***Composição da Aplicação***, é usado nos primeiros estágios da Engenharia de *Software*, quando a prototipagem das interfaces com o usuário, a consideração da interação do *software* com o sistema, a avaliação do desempenho e o julgamento da maturidade tecnológica são de extrema importância. Esse modelo pode também ser utilizado em sistemas desenvolvidos com apoio dos ambientes ICASE[3] (*(Integrated Computer-Aided Software Engineering*). Após serem conhecidos os requisitos e arquitetura básicos do sistema, utiliza-se o modelo ***Pré-Projeto***. Quando a especificação está completa e detalhes sobre o *software* a ser desenvolvido são profundamente conhecidos, o modelo utilizado é o ***Pós-Arquitetura***.

Como mencionado anteriormente, para realizar as estimativas para o *software*, o COCOMO II requer a informação de tamanho do mesmo. Três opções diferentes de dimensionamento estão disponíveis: pontos por objeto, pontos de função e linhas de código fonte.

O modelo Composição da Aplicação utiliza o dimensionamento em pontos por objeto, que mede o tamanho do *software* considerando o número de telas, relatórios e componentes que provavelmente serão necessários para construir a aplicação. Os demais modelos utilizam o dimensionamento em pontos de função ou linhas de código fonte, pois contam com informações mais precisas e detalhadas do *software*, possibilitando, assim, a utilização dessas técnicas.

O primeiro passo para realizar as estimativas utilizando o COCOMO II é determinar o modelo que será utilizado. Em seguida, o tamanho do *software* deve ser dimensionado e as fórmulas de cálculo do esforço particulares a cada modelo devem ser utilizadas.

[3] Ambientes que geralmente incluem as seguintes capacidades: um *framework* para aplicações com *middleware* para integrar e gerenciar a execução dos componentes da aplicação; um conjunto de utilitários comuns, tais como construtores de interfaces gráficas para o usuário, sistemas gerenciadores de bancos de dados e pacotes de suporte à rede; um conjunto de componentes reutilizáveis por domínio; um repositório capaz de gerenciar e permitir acesso a esses componentes; e ferramentas de desenvolvimento para projeto, implementação, integração e testes.

Para calcular o esforço, os modelos Pré-Projeto e Pós-Arquitetura utilizam uma variedade de fatores de equilíbrio e direcionadores de custos para ajustar o cálculo das estimativas. Os fatores de equilíbrio são cinco fatores que são analisados para considerar as principais peculiaridades do projeto que influenciam na relação tamanho x esforço, como por exemplo o grau de similaridade do projeto em desenvolvimento com relação a outros desenvolvidos anteriormente. Os direcionadores de custos são características que devem ser avaliadas em um sistema, pois influenciam diretamente no esforço e prazo necessários para desenvolvê-lo. Um exemplo dessas características é a capacidade dos analistas e programadores envolvidos no projeto. Cada característica analisada, ou seja, cada direcionador e cada fator de equilíbrio, recebe um nível de influência para o projeto (extremamente baixo, muito baixo, baixo, médio, alto, muito alto, extremamente alto). Cada nível tem um valor numérico correspondente (chamado multiplicador de esforço para os direcionadores de custos) que é utilizado nas relações matemáticas do modelo. São esses valores numéricos, juntamente com outras constantes utilizadas nas fórmulas, que tornam o modelo adequado ao projeto em que está sendo aplicado.

O modelo Pré-Projeto considera sete direcionadores de custos e o modelo Pós-Arquitetura considera dezessete. A diferença no número de direcionadores considerados para cada modelo se dá devido às poucas informações conhecidas sobre o projeto nos momentos iniciais e ao detalhamento destas à medida em que o sistema é desenvolvido.

Calculado o esforço para a construção da aplicação, o próximo passo consiste em estimar o prazo necessário. Para realizar esse cálculo, o COCOMO II considera em suas fórmulas o esforço obtido anteriormente e restrições de tempo presentes no desenvolvimento do produto.

Calculados o esforço e o prazo, a equipe média e o custo para o desenvolvimento do projeto podem ser derivados.

Uma importante diferença entre o COCOMO II e o COCOMO 81 é que o modelo atual pode ter os valores de seus direcionadores de custos, fatores de equilíbrio e constantes alterados. Esses valores podem ser alterados por uma nova calibragem realizada pela equipe que propôs o COCOMO II, por uma calibragem desenvolvida por outros grupos de pesquisa ou ainda por calibragem desenvolvida por empresas específicas que utilizem o modelo com base em suas próprias experiências no desenvolvimento de

software. Sendo assim, cada organização pode ser capaz de determinar seus próprios valores, baseando-se em projetos que já tenham sido realizados (TRINDADE *et al.*, 1999).

A calibração do modelo a uma organização específica possibilita a realização de estimativas mais acuradas, uma vez que estarão sendo considerados dados históricos da própria organização para realizar a calibração.

2.4 Considerações Finais

Devido à grande importância das estimativas nos projetos de desenvolvimento de *software* e ao registro de uma considerável responsabilidade pelo fracasso dos mesmos às estimativas mal realizadas, várias técnicas têm sido propostas e muitas delas, com o passar do tempo, têm evoluído com o objetivo de acompanhar as mudanças tecnológicas e de negócio impostas pelo mercado. A escolha da técnica a ser utilizada em um projeto não é tarefa trivial e nenhuma pode garantir valores precisos.

Uma combinação de técnicas paramétricas, como a Análise de Pontos de Função e COCOMO II, e não paramétricas, como a Analogia de Estimativas e Julgamento de Especialistas, tem sido considerada uma boa solução, pois as técnicas paramétricas podem auxiliar nas estimativas iniciais e as não paramétricas podem utilizar dados de projetos já concluídos ou o conhecimento obtido em experiências passadas, buscando uma diminuição na margem de erro das estimativas. As técnicas não paramétricas também são importantes quando a organização não possui experiência para aplicar os modelos paramétricos.

Mesmo considerando-se a falta de precisão das estimativas obtidas, é importante a utilização dessas técnicas, pois à medida que projetos forem concluídos, será possível observar a margem de erros e calibrar o modelo para diminuí-la. A realização *ad hoc* das estimativas ainda é uma escolha das organizações, mas é uma escolha que pode custar muito caro.

Capítulo 3

Ambientes de Desenvolvimento de *Software* Orientados à Organização e a Estação TABA

Este capítulo apresenta os principais conceitos de gerência do conhecimento, de Ambientes de Desenvolvimento de Software Orientados à Organização e apresenta a Estação TABA e suas características relevantes a este trabalho.

3.1 Introdução

O conhecimento tem sido apontado pelas organizações em geral como um importante recurso estratégico, influenciando na competitividade da organização (WINCH, 2000, O'LEARY, 1998, LEE *et al.*, 2001, LIEBOWITZ, 2002).

Os engenheiros de *software* lidam constantemente com diferentes tipos de conhecimento ao longo do processo de desenvolvimento de *software*. O conhecimento acumulado pelos diversos membros da equipe de um projeto pode ser útil em projetos futuros da organização e representa uma potencial fonte de aprendizado organizacional em Engenharia de *Software*. Entretanto, a identificação, organização, armazenamento, utilização, evolução e difusão do conhecimento não são atividades triviais. A gerência do conhecimento surgiu com o objetivo de apoiar essas atividades. Para prover a Estação TABA da capacidade de realizar esse apoio foram definidos os Ambientes de Desenvolvimento de *Software* Orientados à Organização (ADSOrg) (VILLELA *et al.*, 2001a).

Este capítulo apresenta, na seção 3.2, de forma sucinta, os conceitos de gerência do conhecimento, enfatizando-se os pontos relevantes para os Ambientes de Desenvolvimento de *Software* Orientados à Organização e para este trabalho. Em seguida, na seção 3.3, a Estação TABA é apresentada para que seja possível o entendimento da infra-estrutura que permite a configuração de um ADSOrg. Na seção 3.4 são apresentadas as características e a estrutura dos Ambientes de Desenvolvimento de *Software* Orientados à Organização configurados na Estação TABA.

3.2 Gerência do Conhecimento

As organizações defrontam-se com ambientes cada vez mais competitivos. Constantemente surge necessidade de redução de custos, de aumento de valor de mercado e de reestruturações organizacionais que, normalmente, resultam em demissões e consequentes perdas de informações críticas para as organizações. Segundo O'LEARY (1998a) e O'LEARY e STUDER (2001), esses são os principais fatores que levaram as organizações a reconhecerem o valor estratégico do conhecimento. No caso específico de organizações cujo negócio é o desenvolvimento de *software*, segundo WEI *et al.* (2002), pode ser acrescentado a este conjunto de fatores a falta de qualidade com que, muitas vezes, é conduzida a atividade de levantamento de requisitos e a alta volatilidade das plataformas de *hardware* e *software*. Esses são os principais fatores que levaram as organizações a se interessarem pela gerência do conhecimento.

A gerência do conhecimento é uma prática formal de organizar, armazenar e facilitar o acesso e reutilização do conhecimento organizacional, através do uso dos avanços da tecnologia da informação (O'LEARY 1998a). É a administração, de forma sistemática e ativa, dos recursos de conhecimento de uma organização, utilizando tecnologia apropriada e visando fornecer benefícios estratégicos à organização, o que envolve a obtenção de conhecimento relevante a partir de fontes internas e/ou externas disponíveis para a organização, disponibilização e distribuição do conhecimento obtido de forma adequada às necessidades dos usuários, geração de novos conhecimentos e eliminação de conhecimento defasado (VILLELA *et al.*, 2000).

A gerência do conhecimento contribui para o aprendizado individual, organizacional e de equipe, dando suporte à disseminação da informação e à inovação dentro da organização (WINCH, 1999; DIENG, 2000). WINCH (2000) afirma, ainda, que para a vantagem competitiva agregada pelo conhecimento ser de fato sustentável, este conhecimento não pode estar no nível do indivíduo, e sim no nível organizacional.

RAMASUBRAMANIAN e JAGADEESAN (2002) ressaltam que é importante observar que a utilização da gerência do conhecimento em organizações de *software* traz benefícios mais imediatos aos projetos de desenvolvimento de *software* em particular do que à organização como um todo. Os benefícios alcançados através de uma gerência do

conhecimento eficaz em um projeto específico incluem a possibilidade de fornecer respostas rápidas às necessidades dos clientes e o aumento da produtividade da equipe como um todo.

Segundo DIENG (2000), a gerência do conhecimento busca atingir os seguintes objetivos: *(i)* transformar o conhecimento individual em coletivo; *(ii)* dar suporte ao aprendizado e integração de um novo membro em uma organização; *(iii)* disseminar melhores práticas; *(iv)* melhorar os processos corporativos de trabalho, a qualidade e a produtividade dos produtos desenvolvidos; e, *(v)* reduzir tempo de entrega de produtos. No contexto específico da Engenharia de *Software*, a gerência do conhecimento visa apoiar o aperfeiçoamento constante das atividades realizadas ao longo do processo de desenvolvimento de *software*, apoiando a criação e transferência de conhecimentos especificamente relacionados a esse contexto na organização (SEPPÃNEN *et al*., 2002).

As organizações têm adquirido conhecimento através da análise das atividades individuais e dos processos de negócios para aprender com seus sucessos e falhas (PREECE *et al.*, 2001). Pesquisas indicam o crescimento da integração das práticas da gerência do conhecimento com os processos de negócios. O desafio, agora, é prover métodos e ferramentas que criem e capturem o conhecimento para essa integração (O'LEARY e STUDER, 2001).

LEE *et al.* (2001) ressaltam que o verdadeiro sentido da gerência do conhecimento deve ser considerado. A gerência do conhecimento não é simplesmente um problema de montar grupos e equipes de aprendizado ou instalar um sistema de gerência de documentos eletrônicos para disseminar as informações. Ao invés disso, é um deslocamento do paradigma de gerência, que envolve pessoas e outros recursos tais como estrutura organizacional, cultura, tecnologia da informação e outros.

3.2.1 Gerência do Conhecimento em Engenharia de Software

O desenvolvimento de *software* é uma área que envolve muitas pessoas em fases e atividades diferentes e que sofre modificações em um espaço de tempo consideravelmente pequeno. A disponibilidade dos recursos normalmente não cresce proporcionalmente ao

crescimento das necessidades e não diminuir a produtividade é uma necessidade. O conhecimento em Engenharia de *Software* é muito extenso e aumenta cada vez mais. As organizações têm problemas para identificar o conteúdo, a localização e o uso do conhecimento. Uma melhoria no uso desse conhecimento é a motivação básica que guia a gerência do conhecimento em Engenharia de *Software*.

Além da motivação básica, RUS e LINDVALL (2002) citam algumas necessidades que também motivam a utilização da gerência do conhecimento em Engenharia de *Software*: *(i)* diminuição do tempo e custo de desenvolvimento e aumento da qualidade; (*ii)* melhoria no processo de tomada de decisões; *(iii)* aquisição de conhecimento sobre novas tecnologias; *(iv)* aquisição de conhecimento sobre novos domínios; *(v)* compartilhamento do conhecimento sobre a política, práticas e cultura organizacionais; *(vi)* conhecimento sobre 'quem sabe o quê' na organização; e, *(vii)* compartilhamento do conhecimento adquirido no desenvolvimento de *software*.

A gerência do conhecimento tem diversas abordagens na Engenharia de *Software*, entre elas a melhoria do processo (SCHNEIDER *et al.*, 2002; RUS e LINDVALL, 2002), a introdução de novas tecnologias e o aumento da qualidade e produtividade. Para realizar a melhoria do processo, o conhecimento organizacional e o conhecimento produzido na execução das atividades diárias do processo devem ser considerados. Segundo SEPPÃNEN *et al.* (2002), utilizar a gerência do conhecimento na melhoria do processo é muito importante, pois as técnicas de Engenharia de *Software* e de Gerência da Qualidade falham se não forem baseadas no conhecimento necessário e existente em uma organização de *software*. Para aumentar a produtividade, é preciso melhorar a capacidade da equipe de Engenharia de *Software* para que esta realize suas tarefas garantindo que o conhecimento adquirido durante o projeto não seja perdido e que o conhecimento armazenado na organização seja utilizado (RUS e LINDVALL, 2002).

RUS e LINDVALL (2002) e SCHNEIDER *et al.* (2002) ressaltam, ainda, a importância da gerência do conhecimento no planejamento de projetos. As organizações podem utilizar a gerência do conhecimento como uma estratégia de prevenção e mitigação de riscos, uma vez que ela apresenta os riscos que poderiam ser ignorados, como por exemplo: repetição de erros e retrabalho devido ao esquecimento de lições aprendidas em projetos anteriores, indisponibilidade de indivíduos que possuem conhecimento importante

e perda do conhecimento com posterior perda de tempo para readquiri-lo. Para o planejamento dos custos e cronograma de projetos, os autores consideram que a utilização do conhecimento organizacional é muito importante na realização das estimativas, podendo ser utilizados modelos analíticos que analisam dados quantitativos de projetos anteriores e propõem estimativas para o projeto corrente. As atividades de acompanhamento do projeto, tais como a comparação entre as estimativas de custo, esforço e cronograma propostos e os valores atuais do projeto, criam o conhecimento do projeto particular. Seus resultados são úteis para o próprio projeto e podem, também, ser base para a criação do conhecimento organizacional de projetos e para o aprendizado, podendo ser armazenados em repositórios e bases de experiências. Utilizando o conhecimento organizacional, os gerentes de projeto podem também estimar os defeitos, confiabilidade e outros parâmetros do projeto e do produto.

REIFER (2002) também faz uma análise dos principais benefícios da gerência do conhecimento para o planejamento do projeto. O autor também considera que a principal vantagem é a captura do conhecimento organizacional e de lições aprendidas e a disponibilização destes de forma sistemática, porém ressalta que capturar as lições aprendidas pode ser uma tarefa difícil e que podem haver poucos incentivos para a utilização da gerência do conhecimento quando há pressões para concluir a fase de planejamento.

Outras atividades da Engenharia de *Software* que são apoiadas pela gerência do conhecimento são a gerência de documentação, a gerência de competências e identificação de especialistas e a reutilização de *software* (RUS e LINDVALL, 2002).

SEPPÃNEN *et al.* (2002) realizaram um estudo sobre como utilizar a gerência do conhecimento através dos processos de projetos de *software*. Para isso, realizaram uma pesquisa em uma unidade de negócios independente de uma empresa de desenvolvimento de *software* para produtos eletrônicos. Os autores propuseram uma abordagem que captura o conhecimento de fontes relevantes ao projeto e, em seguida, empacota o conhecimento e o disponibiliza aos membros do projeto para utilização, de acordo com a demanda. A captura, neste caso, é similar à análise organizacional na Fábrica de Experiências, onde é feita a análise do conhecimento e o empacotamento deste em blocos reutilizáveis. A

abordagem dos autores trata do conhecimento adquirido ao longo do processo de desenvolvimento de *software*.

RAMESH (2002) ressalta que para utilizar a gerência do conhecimento em processos de desenvolvimento de *software* é necessário que haja rastreabilidade no processo de gerência do conhecimento para auxiliar a conexão dos fragmentos de conhecimento capturados ao longo do processo de desenvolvimento.

BIRK *et al.* (2002) consideram a Análise *PostMortem* como um excelente método para gerência do conhecimento, pois captura a experiência e melhores práticas de projetos concluídos, com o objetivo de que os membros dos projetos sejam capazes de reconhecer, recordar e utilizar esse aprendizado durante os novos projetos.

3.2.2 Conhecimento

Como mencionado, a gerência do conhecimento é um campo que vem ganhando popularidade tanto na comunidade acadêmica quanto no mundo dos negócios. Entretanto, embora exista uma certa abundância em propostas de dispositivos e métodos para desenvolver sistemas baseados em conhecimento, ainda existe uma confusão sobre o que constitui conhecimento (BIGGAN, 2001).

Conceitualmente, conhecimento é a informação combinada com experiência, contexto, interpretação e reflexão. É a forma de informação que está pronta para ser aplicada em decisões e ações (DAVENPORT *et al.*, 1998).

FISCHER e OSTWALD (2001) enfatizam ainda que conhecimento é muito mais do que informação. Conhecimento é a informação inserida em um contexto particular. Uma informação pode facilmente tramitar de um lugar para outro, de uma pessoa para outra, mas o conhecimento não o pode fazer com tal facilidade.

ALAVI e LEIDNER (1999), por outro lado, preferem explicitar a conexão entre informação e conhecimento, diminuindo a distância entre seus conceitos. Afirmam que o conceito de conhecimento não é radicalmente diferente de informação, uma vez que conhecimento é informação processada na mente do indivíduo, que volta a se tornar informação quando é articulado ou comunicado a outros através de textos, saídas computacionais, palavras faladas ou escritas e outros meios.

Segundo BIGGAN (2001), um cuidado especial deve ser tomado para diferenciar conhecimento de crença e opinião. Para isso, é necessário que três critérios sejam satisfeitos: *(i)* o conhecimento deve ser verdadeiro; *(ii)* aquele que detém o conhecimento deve acreditar que ele é verdadeiro; e *(iii)* aquele que detém o conhecimento deve estar em posição de saber que o mesmo é verdadeiro.

Quando se trata de conhecimento, existe uma unanimidade no que diz respeito aos seus dois tipos, sendo eles explícito e tácito. Conhecimento explícito pode ser expresso em palavras e números e pode ser facilmente transmitido e compartilhado na forma de princípios universais, fórmulas científicas, procedimentos codificados, entre outros. Conhecimento tácito é altamente pessoal e difícil de formalizar, tornando-se difícil de compartilhar com outras pessoas. Conhecimento tácito depende da experiência pessoal e envolve fatores intangíveis como crenças, perspectivas e valores. Intuição e previsões pertencem a esta categoria de conhecimento (VILLELA *et al.*, 2000).

Segundo RUS e LINDVALL (2002) o principal benefício do conhecimento explícito é que ele pode ser armazenado, organizado e disseminado sem o envolvimento de quem originou esse conhecimento.

Alguns autores citam outras classificações ou subclassificações para o conhecimento (BIGGAN, 2001; MARKKULA, 1999).

Segundo MARKKULA (1999), as metas organizacionais determinam o tipo de conhecimento a ser coletado por uma organização, sendo crítica a identificação e coleta de conhecimento independente de sua forma. O autor classifica os conhecimentos que podem ser úteis para organizações que desenvolvem e mantêm *software*, separando-os em três classes: conhecimento externo, conhecimento interno estruturado e conhecimento interno informal.

Na classe de conhecimento externo, tem-se: *(i)* ponteiros para *sites* da Internet, principalmente os *sites* técnicos, mas também podem ser *sites* de clientes, competidores, parceiros técnicos, fornecedores de *software*, jornais técnicos e centros de pesquisa, e *(ii)* manuais, livros e materiais de treinamento disponíveis eletronicamente ou a informação de como obtê-los.

A classe de conhecimento interno estruturado inclui: métodos de planejamento de projetos e de Engenharia de *Software*, modelos de documentos com exemplos reais de

utilização, melhores práticas, componentes de *software*, relatórios de pesquisa, diretrizes e outras informações de comunicações específicas da organização, competência dos funcionários, informações de marketing e de resultados da organização. Estes tipos de conhecimento são revisados pela organização antes de serem publicados.

Por fim, a classe de conhecimento interno informal é dividida em três partes: discussões, que podem ser organizadas de acordo com os diferentes assuntos; notícias relacionadas a assuntos técnicos e pastas de projeto, onde são mantidos todas as informações e todo o conhecimento gerados durante cada projeto e que servem como repositórios de lições aprendidas quando os projetos são concluídos.

3.2.3 Memória Organizacional

A Memória Organizacional pode ser considerada como um repositório do conhecimento disponível na organização, cuja finalidade é assegurar que o conhecimento desejado possa ser recuperado no tempo e no lugar corretos (KOUWENHOVEN, 1998).

Para DIENG (2000), a Memória Organizacional é uma representação explícita e persistente da informação e conhecimento cruciais em uma organização, de forma a facilitar seu acesso, compartilhamento e reuso pelos membros da organização para executar tarefas individuais e coletivas.

A Memória Organizacional tem dois papéis principais, segundo FISCHER e OSTWALD (2001): *(i)* ser fonte de informação para auxiliar a organização a entender seus problemas e *(ii)* ser receptáculo de novas informações e produtos criados durante o desenvolvimento das tarefas.

Quando se diz receptáculo ou repositório é importante ressaltar que não pode ser entendido como um repositório físico único, monolítico e de algum tipo específico, como observado por ACKERMAN e HALVERSON (2000), pois nem mesmo as organizações são entidades monolíticas.

Segundo O'LEARY (1998a), a Memória Organizacional pode conter várias bases de conhecimento, que podem ser para utilização pela máquina ou pelo usuário, sendo bases de conhecimento típicas em empresas de *software*: propostas, projetos, melhores práticas, notícias, especialistas, entre outras.

O'LEARY (1998b, 1998c) acredita que, para usar várias bases de conhecimento de forma efetiva, as organizações devem estar hábeis a gerar ontologias que permitam aos usuários estabelecer os recursos que precisam e desejam. As ontologias[4] e as bases de conhecimento estão proximamente relacionadas na gerência do conhecimento, pois as ontologias definem as características das bases de conhecimento e suas visões, além de facilitarem a comunicação entre múltiplos usuários e a associação entre as múltiplas bases de conhecimento, pois especificam um vocabulário compartilhado.

3.2.4 Processo de Gerência de Conhecimento

Para implantar gerência do conhecimento é necessário que um processo seja seguido. PREECE *et al.* (2001) afirmam que poucas organizações possuem um processo sistemático de gerência do conhecimento. No entanto, pesquisadores têm proposto diversos processos de conhecimento, sendo que, apesar de possuírem fases ou estágios diferentes, todos apontam para as mesmas fases genéricas: a criação, captura e disponibilização do conhecimento. A diferença entre os processos propostos está no nível de detalhamento das atividades.

JOSHI (2001) propõe um processo de conhecimento composto pelas seguintes atividades genéricas de conhecimento: seleção de conhecimento, aquisição de conhecimento, utilização de conhecimento, transferência de conhecimento e internalização de conhecimento. A seleção de conhecimento refere-se à atividade de identificar e recuperar o conhecimento necessário dentre os recursos de conhecimento existentes na organização. A seleção de conhecimento e aquisição de conhecimento são atividades análogas, diferindo em um importante aspecto: a aquisição de conhecimento manipula o conhecimento que existe também fora dos limites da organização. A utilização de conhecimento é a atividade de aplicação do conhecimento existente para gerar novo

[4] Ontologia é uma representação de vocabulário, geralmente especializada para algum domínio ou assunto. O termo ontologia algumas vezes é utilizado para se referir a um corpo de conhecimento, tipicamente um senso comum sobre um determinado domínio, utilizando um vocabulário como representação. Em outras palavras, a representação textual provê um conjunto de termos com os quais são descritos os fatos em algum domínio, enquanto o corpo do conhecimento utilizando aquele vocabulário é uma coleção de fatos sobre um domínio (CHANDRASEKARAN et al., 1999).

conhecimento e/ou produzir uma externalização do conhecimento. A transferência de conhecimento refere-se à disseminação e distribuição do conhecimento. A internalização de conhecimento refere-se a uma atividade que altera os recursos de conhecimento da organização através da aquisição, seleção, transferência e/ou utilização do conhecimento.

Outras propostas de processos de conhecimento são encontradas na literatura, entre elas: *(i)* RUS e LINDVALL (2002) consideram as seguintes atividades no processo: criação, captura, transformação, acesso e aplicação; *(ii)* PREECE *et al.* (2001) propõem um processo com as fases de análise de requisitos, modelagem conceitual, construção de base de conhecimento, validação e implantação, refinamento e manutenção; *(iii)* STAAB *et al.* (2001) propõem um processo de conhecimento formado por quatro passos: criação ou importação do conhecimento, captura do conhecimento, recuperação e acesso do conhecimento, utilização do conhecimento; *(iv)* FISCHER e OSTWALD (2001) consideram a gerência do conhecimento como um processo cíclico que envolve criação, integração e disseminação; e *(v)* LEE *et al.* (2001) propõem um processo de gerência do conhecimento com quatro estágios: início, propagação, integração e distribuição.

3.3 A Estação TABA

A Estação TABA, conforme originalmente proposto, é um meta-ambiente capaz de gerar, através de instanciação, ambientes de desenvolvimento de *software* (ADS) adequados às particularidades de processos de desenvolvimento e de projetos específicos. Um meta-ambiente pode ser definido como um ambiente que abriga um conjunto de programas que interage com usuários para definir interfaces, selecionar ferramentas e estabelecer os tipos de objetos que irão compor o ambiente de desenvolvimento específico (ROCHA *et al.*, 1990).

O projeto TABA teve início em 1990, a partir da constatação de que domínios de aplicação diferentes possuem características distintas, e que estas devem incidir nos ambientes de desenvolvimento através dos quais os engenheiros de *software* desenvolvem aplicações. Sendo assim, a Estação TABA tem por objetivo auxiliar na definição, implementação e execução de ADS adequados a contextos específicos.

Com o intuito de atender a esse objetivo, quatro funções foram originalmente definidas para a Estação TABA (TRAVASSOS, 1994): *(i)* auxiliar o engenheiro de

software na especificação e instanciação do ambiente mais adequado ao desenvolvimento de um produto específico a partir do processo de *software* e/ou de uma definição do domínio de aplicação; *(ii)* auxiliar o engenheiro de *software* na implementação das ferramentas necessárias ao ambiente definido; *(iii)* permitir aos desenvolvedores do produto de *software* a utilização da Estação através do ambiente instanciado; e *(iv)* permitir a execução do *software* na Estação configurada para o seu desenvolvimento.

Considerando que os objetivos iniciais do projeto, os quais apontavam que as particularidades de domínios de aplicação também devem ser consideradas na instanciação de um ADS para que este ofereça ferramentas mais específicas e possibilite a reutilização do conhecimento do domínio, OLIVEIRA (1999) estendeu a definição da Estação TABA de forma que esta possa instanciar ADS orientados a domínios específicos, criando os Ambientes de Desenvolvimento de *Software* Orientados a Domínio (ADSOD), através da inclusão do conhecimento de domínio e sua disponibilização para a realização das diversas atividades do desenvolvimento de *software*.

3.4 Ambientes de Desenvolvimento de *Software* Orientados à Organização (ADSOrg)

Para melhorar a maneira como as organizações desenvolvem e mantêm *software* é fundamental melhorar a maneira como elas administram o conhecimento requerido para a realização dessa atividade. Dessa forma, a partir dos estudos realizados em gerência do conhecimento e ADS, foi definido o conceito de Ambiente de Desenvolvimento de *Software* Orientado à Organização (ADSOrg), uma classe de ADS que apóia a atividade de Engenharia de *Software* em uma organização, fornecendo o conhecimento acumulado pela organização e relevante para essa atividade, ao mesmo tempo em que apóia, a partir dos projetos específicos, o aprendizado organizacional em Engenharia de *Software* (VILLELA, *et al.* ,2000).

ADSOrg representam uma evolução dos Ambientes de Desenvolvimento de *Software* Orientados a Domínio (ADSOD) e visam apoiar o gerenciamento completo do conhecimento requerido em uma atividade de Engenharia de *Software*, evitando que esse conhecimento fique disperso ao longo da estrutura organizacional e, consequentemente, sujeito a dificuldades de acesso e, até mesmo, a perdas (VILLELA *et al.*, 2000).

Um ADSOrg tem os seguintes objetivos: *(i)* permitir que desenvolvedores e demais envolvidos no desenvolvimento de *software* tenham acesso a todo conhecimento acumulado pela organização e relevante ao contexto de desenvolvimento e manutenção de *software*; *(ii)* promover o aprendizado organizacional neste contexto (VILLELA *et al.*, 2000; VILLELA *et al.*, 2001b; VILLELA *et al.*, 2002).

Um ADSOrg torna possível a identificação de erros cometidos em projetos anteriores e a reutilização de soluções pré-qualificadas durante a realização de tarefas similares ou idênticas, visando a melhoria da produtividade e qualidade, bem como a redução dos custos do *software* (VILLELA *et al.*, 2000).

ADSOrg são, como mencionado anteriormente, uma evolução do conceito de ADSOD e foram construídos a partir de duas constatações: *(i)* duas ou mais organizações podem desenvolver *software* para um mesmo domínio com processos, interesses e características muito distintas; e, *(ii)* o conhecimento do domínio não é o único conhecimento importante para apoiar desenvolvedores e demais envolvidos no processo de *software* em suas atividades. Outros conhecimentos, tais como o conhecimento relativo às diretrizes e melhores práticas organizacionais e às lições aprendidas em experiências anteriores com o uso de processos, métodos e técnicas de *software*, também são extremamente importantes e úteis para os desenvolvedores e demais envolvidos.

Os requisitos de um ADSOrg são: *(i)* possuir a representação da estrutura e dos processos organizacionais e possibilitar a fácil localização de especialistas cujo conhecimento e experiência podem ser úteis em projetos de *software*; *(ii)* armazenar conhecimento especializado sobre desenvolvimento e manutenção de *software* e fornecer esse conhecimento para as equipes de projeto quando necessário; e, *(iii)* apoiar a contínua evolução do conhecimento armazenado no ambiente.

Dessa forma, a Estação TABA passou a ter as seguintes funções VILLELA *et al.*, 2000):

(i) Auxiliar o engenheiro de *software* na especificação e configuração do ambiente mais adequado para apoiar o desenvolvimento de *software* em uma organização específica (ADSOrg), considerando seu processo de *software* e a gerência do conhecimento organizacional relevante para o desenvolvimento de *software*;

(ii) Auxiliar o engenheiro de *software* na especificação e instanciação de ambientes de desenvolvimento de *software* para projetos específicos a partir da própria Estação TABA (caso não seja possível ou adequado a definição de um ADSOrg);

(iii) Auxiliar os gerentes de projeto de organizações cujo negócio é o desenvolvimento de *software* para diversos clientes na realização de novas especializações a partir dos processos especializados do ADSOrg, de forma a atender às particularidades de desenvolvimento de *software* desses clientes;

(iv) Auxiliar os gerentes de projeto na especificação e instanciação de ambientes de desenvolvimento de *software* para projetos específicos a partir do ADSOrg configurado;

(v) Auxiliar o engenheiro de *software* a implementar ferramentas necessárias aos ambientes de desenvolvimento de *software*;

(vi) Permitir o uso da Estação TABA para desenvolvimento de *software* através dos ambientes instanciados; e

(vii) Permitir a execução do *software* na própria Estação.

3.4.1 Planejamento de Projetos em Ambientes de Desenvolvimento de Software Orientados à Organização

Para apoiar a atividade de Planejamento de Projetos na Estação TABA, algumas ferramentas foram desenvolvidas e outras encontram-se em desenvolvimento, no contexto do ADSOrg.

Uma delas, *RiscPlan*, apóia o planejamento de riscos baseado na reutilização do conhecimento organizacional de riscos, considerando as características do projeto e a experiência da organização em projetos anteriores (FARIAS, 2002). O objetivo dessa ferramenta é dar suporte automatizado ao planejamento de riscos em projetos de *software*, oferecendo aos gerentes de projeto o conhecimento e experiências acumulados por diferentes gerentes em projetos similares ocorridos dentro da própria organização.

Outra ferramenta, *RHPlan*, apóia as atividades de definição de perfis de competência,

seleção de profissionais, monitoração da alocação e avaliação de recursos humanos necessárias ao processo de alocação de recursos humanos de um projeto (SCHNAIDER, 2003). O objetivo dessa ferramenta é dar suporte automatizado ao planejamento de recursos humanos em projetos de *software,* permitindo a utilização do conhecimento organizacional de competências.

A ferramenta *CustPlan,* proposta neste trabalho, auxilia o planejamento de custos e prazos em ADSOrg, considerando as características do projeto, a experiência da organização em projetos anteriores e métodos paramétricos de realização de estimativas. O objetivo dessa ferramenta é dar suporte automatizado ao planejamento de custos e prazos em projetos de *software*, oferecendo aos gerentes de projeto o conhecimento e experiências acumulados por diferentes gerentes em projetos similares ocorridos dentro da própria organização.

A ferramenta *CustPlan* está inserida no contexto do ADSOrg e, assim, deve atender a seus requisitos. Dessa forma, o requisito *(ii)* que designa "fornecer conhecimento para as equipes de projeto" e o requisito *(iii)* que designa "apoiar a contínua evolução do conhecimento armazenado no ambiente" serão parcialmente atendidos no que diz respeito ao conhecimento necessário para a realização das atividades dos processos de gerência de prazos e custos, uma vez que, durante o planejamento de prazos e custos de um projeto será possível a reutilização do conhecimento adquirido ao longo de projetos anteriores e armazenado na memória organizacional.

A utilização das ferramentas descritas acima fornece o Plano de Riscos, o Plano de Alocação de Recursos Humanos, o Plano de Custos e o Cronograma do projeto. Esses planos são componentes do Plano do Projeto que é utilizado na gerência do projeto.

3.5 Considerações Finais

Este capítulo apresentou os principais conceitos da gerência do conhecimento, importantes para a definição dos Ambientes de Desenvolvimento de *Software* Orientados à Organização.

Também foi realizada a apresentação das principais características da Estação TABA, relevantes para o ADSOrg e para este trabalho.

O ADSOrg foi definido e foram descritas suas principais características. As ferramentas de apoio ao planejamento de projeto, presentes no ADSOrg, foram apresentadas.

A utilização do conhecimento organizacional é uma das principais atividades em um ADSOrg. A ferramenta proposta neste trabalho contribui para a formação desse conhecimento no que diz respeito às atividades dos processos de gerência de custos e prazos dos projetos de desenvolvimento de *software* da organização e utiliza o conhecimento armazenado para auxiliar a realização de melhores estimativas de custos e prazos no Cronograma e Plano de Custos dos projetos a serem desenvolvidos.

No próximo capítulo, será apresentada a abordagem proposta para o planejamento de prazos e custos para projetos de *software* em ADSOrg. Com esse objetivo serão apresentadas a abordagem teórica para processos de gerência de custos e gerência de tempo e suas abordagens no ADSOrg.

Capítulo 4

Planejamento de Custos em Ambientes de Desenvolvimento de *Software* Orientados à Organização

Este capítulo apresenta uma abordagem para o planejamento de custos em projetos de software baseada na reutilização do conhecimento e experiência organizacionais. A ferramenta proposta introduz o planejamento de custos à Estação TABA e é fundamentada nos conceitos de gerência do conhecimento e ADSOrg.

4.1 Introdução

O planejamento de custos é uma das atividades iniciais de um projeto de *software* e requer uma visão global da organização, sendo fortemente centrado na experiência e conhecimento adquiridos em projetos anteriores. Quanto maior a experiência do gerente do projeto, melhor ele será capaz de realizar as estimativas para o projeto corrente. Porém, o conhecimento de prazo e custos de um gerente de projeto não pode permanecer no nível do indivíduo. Para que a organização evolua aprendendo com seus próprios erros e acertos, é necessário que o conhecimento seja gerenciado de forma a tornar possível sua captura, recuperação e futura utilização.

Este capítulo apresenta uma abordagem para o planejamento de custos e fundamenta-se nos conceitos de gerência do conhecimento e Ambientes de Desenvolvimento de *Software* Orientados à Organização. A proposta é disponibilizar ao gerente do projeto todo o conhecimento de prazos e custos acumulado pelos vários gerentes de uma organização, considerando projetos similares anteriores.

A seção seguinte apresenta como o planejamento de projetos será realizado em um ADSOrg e aborda o planejamento de custos como uma das atividades realizadas durante o planejamento do projeto. A seção 4.3 apresenta os processos de gerência de tempo e

gerência de custos definidos buscando-se a utilização do conhecimento organizacional de tempo e custos nas várias atividades do processo. A seção 4.4 apresenta a abordagem proposta para o planejamento de custos, apresentando a modelagem das várias atividades incluídas nos processos de gerência de tempo e gerência de custos e a forma como o conhecimento é utilizado e, finalmente, a seção 4.5 apresenta as considerações finais.

4.2 Planejamento de Custos em ADSOrg

A Estação TABA permite que uma organização estabeleça seu Processo Padrão e, a partir dele, defina os processos especializados e instanciados. Um processo instanciado se adequa às necessidades de um projeto específico e precisa ser acompanhado, medido e gerenciado de forma a buscar uma maior qualidade do produto de *software* que está sendo desenvolvido.

Uma das atividades que deve estar presente em um processo instanciado é a atividade de *Planejamento do Projeto*. Essa atividade deve ser realizada no início do projeto, de forma macroscópica, e deve ser detalhada ao longo do desenvolvimento, à medida em que aumenta o nível de entendimento do projeto. A atividade de planejamento tem como produto o documento Plano do Projeto, que contém as diretrizes e métodos a serem seguidos para a execução do projeto. O Plano do Projeto é composto por várias seções que abordam tópicos específicos a serem considerados. O Plano do Projeto elaborado com o auxílio de um ADSOrg conterá os seguintes planos:

Plano de Organização: descreve as equipes de desenvolvimento, gerência e controle da qualidade que atuarão no projeto e respectivas responsabilidades.

Plano de Documentação: apresenta os roteiros de todos os documentos a serem gerados ao longo do desenvolvimento do projeto, bem como as responsabilidades por sua produção e avaliação.

Plano de Recursos e Produtos: identifica os produtos a serem gerados, respectivas dimensões e os recursos necessários para o desenvolvimento do projeto. É neste plano que encontra-se a análise dos prazos e recursos humanos para a realização das atividades do projeto.

Plano de Acompanhamento: descreve os marcos e pontos de controle gerenciais e os procedimentos adotados para acompanhamento e controle do projeto. Este plano contém, ainda, a análise de riscos do projeto e os procedimentos de gerência relativos à mesma.

Plano de Controle da Qualidade: descreve os marcos e pontos de controle da qualidade e todos os procedimentos (revisões) e atributos de qualidade a serem adotados para controle da qualidade ao longo do desenvolvimento e avaliação da qualidade do produto final.

Plano de Treinamento: descreve o cronograma das atividades para treinamento dos desenvolvedores e usuários do produto.

Plano de Implantação e Operação: descreve os procedimentos necessários para a implantação e operação do produto.

Plano de Gerência de Configuração: descreve os procedimentos para gerência de configuração.

A figura 4.1 ilustra um modelo do Plano do Projeto a ser gerado em um ADSOrg.

Plano do Projeto

1. **Plano de Organização**
2. **Plano de Documentação**
3. **Plano de Recursos**

   4.1 Recursos Humanos
   (indica os recursos humanos necessários ao desenvolvimento do projeto)

   4.2 Cronograma
   (indica o cronograma para a realização do projeto)

   4.3 Recursos de Hardware
   (indica os recursos de hardware necessários ao desenvolvimento do projeto)

   4.4 Recursos de Software
   (indica os recursos de software necessários ao desenvolvimento do projeto)

   4.5 Recursos Financeiros
   (indica os recursos financeiros necessários ao desenvolvimento do projeto, representados através do orçamento do projeto)

   4.6 Outros Recursos
   (indica outros recursos necessários ao desenvolvimento do projeto, como por exemplo aluguel, energia, dentre outros)

4. **Plano de Acompanhamento**
5. **Plano de Controle da Qualidade**
6. **Plano de Treinamento**
7. **Plano de Implantação e Operação**
8. **Plano de Gerência da Configuração**

Figura 4.1. Modelo do Plano de Projeto com detalhamento do Plano de Recursos (ROCHA, 2000)

Para fornecer apoio à elaboração do Plano do Projeto em um ADSOrg instanciado, algumas ferramentas foram desenvolvidas e estão disponíveis para a execução da atividade de Planejamento do Projeto (ou equivalente). O objetivo é apoiar o desenvolvimento de cada um dos planos (seções) inseridos no Plano do Projeto, fornecendo para isso, o conhecimento organizacional adquirido pelos vários gerentes de projetos anteriores e que possa ser útil à elaboração de um plano específico. Algumas dessas ferramentas foram citadas no capítulo 3, sendo a *RiscPlan* (FARIAS 2002) utilizada para apoiar a Análise de Riscos e a *RHPlan* (SCHNAIDER, 2003) utilizada para apoiar a alocação de recursos humanos ao projeto.

Neste trabalho apresentamos uma abordagem para o planejamento de custos (cronograma, recursos de *hardware, software*, outros recursos e orçamento) , presente no Plano de Recursos, em ADSOrg, e propomos a *CustPlan* como ferramenta de apoio à abordagem proposta. Com a utilização da *CustPlan* o gerente de projeto poderá elaborar o Plano de Custos utilizando técnicas paramétricas de estimativas e consultando o conhecimento acumulado para a realização dessa atividade.

4.3 Processo de Gerência de Tempo e Processo de Gerência de Custos

Os processos de gerência de tempo e gerência de custos possuem um conjunto de atividades maior que o necessário ao planejamento de tempo e custos de um projeto de *software*, uma vez que abrangem também as atividades relacionadas ao acompanhamento e controle do Plano de Custos ao longo do processo de desenvolvimento. O objetivo da definição do processo foi destacar os requisitos a serem alcançados pela abordagem de planejamento de custos proposta neste trabalho e facilitar o entendimento da inserção da abordagem na gerência de tempo e custos como um todo.

Os processos foram definidos tendo como base a literatura de gerência de custos, as recomendações da norma NBR ISO 10006 (2000), que define diretrizes para a qualidade no gerenciamento de projetos, o relatório técnico 16326 da ISO/IEC (1999), que provê um guia para a aplicação da norma NBR ISO/IEC 12207 (1998) à gerência de projetos de *software*, e o PMBOK (*Project Management Body of Knowledge* - 2000*)*, o padrão para gerência de projetos publicado pelo PMI (*Project Management Institute).*

A norma NBR ISO 10006 (2000) recomenda que sejam utilizados, em todas as atividades do processo, a experiência e dados históricos provenientes de projetos anteriores. Dessa forma, o processo aqui descrito busca a reutilização do conhecimento e experiência organizacionais, um dos benefícios visados pela gerência do conhecimento.

Em todas as referências citadas, a gerência de custos é realizada através de dois pilares de gerência: o tempo do projeto e os custos propriamente ditos. Sendo assim, foram definidos dois processos distintos, que relacionam-se diretamente: *Processo de Gerência de Tempo* e *Processo de Gerência de Custos.*

A descrição dos processos apresenta os aspectos que devem ser considerados, os produtos gerados e os responsáveis envolvidos em cada atividade. Esse processo ainda pode ser adaptado à realidade de uma organização específica, considerando os documentos a serem gerados, realização de sub-atividades e possíveis limitações ou restrições impostas.

4.3.1 Processo de Gerência de Tempo

A *Gerência de Tempo* tem como objetivo determinar as dependências e a duração das atividades do projeto.

Este processo deve ser utilizado em dois momentos distintos: inicialmente, quando pouco é conhecido do projeto, onde são geradas as estimativas iniciais ainda com uma abrangência macroscópica, para realizar o planejamento do projeto e, posteriormente, quando mais informações do projeto forem obtidas, o que permite o refinamento das estimativas geradas.

No momento de realizar as primeiras estimativas, apenas as atividades do processo de desenvolvimento são analisadas. No refinamento das estimativas, as sub-atividades do processo também são consideradas.

O processo de gerência de tempo é composto por cinco atividades:

i. Identificar as dependências entre as atividades do projeto
ii. Estimar a duração das atividades do projeto com abordagem *top-down*
iii. Estimar a duração das atividades do projeto com abordagem *bottom-up*
iv. Elaborar o cronograma do projeto
v. Controlar o cronograma do projeto

i. Identificar as Dependências entre as Atividades do Projeto

Esta atividade consiste em identificar as relações de dependência entre as atividades do projeto. As inter-relações, interações lógicas e interdependências entre as atividades devem ser identificadas e analisadas quanto à sua consistência, pois existem atividades que podem ser realizadas independentemente de outras, mas há aquelas que precisam de uma relação de dependência temporal com outra(s).

Durante esta sub-atividade, o gerente do projeto deve identificar as dependências entre as atividades/sub-atividades do processo de desenvolvimento do projeto. A análise das dependências usuais entre as atividades/sub-atividades de projetos anteriormente realizados buscando encontrar informações que orientem na determinação das dependências entre as atividades/sub-atividades do projeto corrente é de grande importância, bem como considerar a opinião de especialistas.

Produto: Atividades/sub-atividades do projeto sequenciadas.

Responsável: Gerente do projeto.

Após esta atividade, são realizadas as estimativas de duração das atividades do projeto e seu esforço. Neste momento, são estimados quantos períodos de trabalho e o esforço que serão necessários para a realização de cada atividade/sub-atividade do projeto. Os períodos de trabalho podem ser indicados em horas, dias, semanas ou outras unidades de tempo.

Ao estimar os períodos de trabalho é preciso considerar também o tempo de espera necessário devido às relações de dependência entre algumas atividades/sub-atividades. Essas relações foram identificadas durante a execução da atividade *i.*

As estimativas de duração das atividades podem ser realizadas com abordagem *top-down* (atividade *(ii)* Estimar a duração das Atividades do Projeto com Abordagem *Top-down*) ou com abordagem *bottom-up* (atividade *(iii)* Estimar a Duração das Atividades do Projeto com Abordagem *Bottom-up*). Dessa forma, as atividades *ii* e *iii* são realizadas de forma alternativa.

ii. Estimar a duração das Atividades do Projeto com Abordagem Top-down

Gerentes de projetos costumam estimar utilizando métodos paramétricos e/ou comparação com projetos similares, de acordo com a própria experiência em projetos anteriores. Neste processo, consideramos que será utilizada uma combinação destes.

Modelos paramétricos serão utilizados para a realização das estimativas do projeto como um todo ou por partes (módulos, fases). Dados de projetos similares anteriores serão utilizados para refinar essas estimativas. Após obtidos esses valores, os mesmos devem ser distribuídos entre as atividades/sub-atividades, sendo muito importante, novamente, consultar dados de projetos anteriormente realizados e considerar a opinião de especialistas.

Sub-atividades:

- *Realizar as estimativas do projeto através de Métodos Paramétricos:* Durante esta sub-atividade o gerente do projeto utiliza métodos paramétricos (Análise de Pontos de Função e/ou COCOMO II) para realizar as estimativas do projeto. O gerente pode realizar as estimativas de todo o projeto ou dividi-lo em módulos e realizar as estimativas para cada módulo. Nesse caso as estimativas totais do projeto são obtidas através do somatório das estimativas dos módulos.
- *Ajustar as estimativas a partir de dados de projetos similares:* Durante esta sub-atividade o gerente do projeto consulta dados de projetos similares para refinar as estimativas do projeto obtidas na sub-atividade anterior. Essas estimativas serão utilizadas como base para realização das estimativas de tempo das atividades/sub-atividades do projeto.
- *Distribuir o tempo do projeto pelas atividades/sub-atividades do processo de desenvolvimento:* Durante esta sub-atividade o gerente do projeto realiza as estimativas de duração de cada atividade/sub-atividade do processo de desenvolvimento. Para realizar essa sub-atividade o gerente distribui os valores totais do projeto ou de seus módulos, encontrados com a utilização dos modelos paramétricos e ajustados com dados de projetos similares (duas sub-atividades descritas acima), entre as atividades/sub-atividades do projeto. Para isso, o gerente deve analisar o tempo das atividades/sub-

atividades de projetos anteriormente realizados buscando encontrar informações que orientem na realização das estimativas de tempo das atividades/sub-atividades do projeto corrente, bem como considerar sua experiência pessoal.

Produto: Atividades/ sub-atividades do projeto com estimativas de tempo.

Responsável: Gerente do projeto.

iii. Estimar a duração das Atividades do Projeto com Abordagem Bottom-up

Caso o gerente faça a opção pela abordagem *bottom-up*, ele realiza as estimativas de cada atividade/sub-atividade do projeto. Os valores totais do projeto e/ou seus módulos são, então, obtidos através do somatório das estimativas das atividades/sub-atividades. É possível perceber que abordagem *bottom-up* ocorre de forma inversa à *top-down*, onde as estimativas são realizadas, inicialmente, para todo o projeto e, depois, distribuídas entre suas atividades/sub-atividades. Aqui, na abordagem *bottom-up*, inicialmente, são realizadas as estimativas das atividades/sub-atividades e, em seguida, são totalizadas as estimativas do projeto. Nesse caso, para estimar o tempo e esforço de cada atividade/sub-atividade, o gerente do projeto deve analisar a duração e esforço das atividades/sub-atividades presentes em projetos similares anteriores, buscando encontrar valores que orientem a realização das estimativas das atividades/sub-atividades do projeto corrente.

Produto: Atividades/ sub-atividades do projeto com estimativas de tempo.

Responsável: Gerente do projeto.

iv. Elaborar o Cronograma do Projeto

Esta atividade consiste em identificar os caminhos críticos do projeto, estabelecer as datas de início e fim das atividades do projeto e registrar os marcos e pontos de controle identificados para o projeto no Plano de Acompanhamento.

Sub-atividades:

- *Identificar caminhos críticos do projeto:* Durante esta sub-atividade o gerente do projeto deve identificar os caminhos críticos do projeto. Um caminho crítico é uma seqüência de atividades tais que, se houver atraso em alguma delas, esse atraso será transmitido ao término do projeto. Para cada

caminho crítico devem ser registradas informações importantes, como a necessidade de tomada de decisão no caminho crítico em questão ou alguma observação a ele pertinente.

- *Determinar as datas de início e fim de cada atividade/sub-atividade:* Durante esta sub-atividade o gerente do projeto define as datas de início e fim para realizar cada atividade/sub-atividade, observando o prazo estimado para cada uma delas, suas dependências e os caminhos críticos identificados.
- *Registrar marcos e pontos de controle do projeto:* Durante esta sub-atividade os marcos e pontos de controle identificados no Plano de Acompanhamento do projeto são registrados no cronograma.

Produto: Cronograma do projeto.

Responsável: Gerente do projeto.

v. Controlar o Cronograma do Projeto

Durante o desenvolvimento do projeto, o cronograma deve ser comparado, analisado e revisto sempre que necessário.

Convém que o cliente e as principais partes interessadas estejam cientes das alterações do cronograma e, quando necessário, sejam envolvidos na determinação das mudanças.

Sub-atividades:

- *Identificar Desvios de Cronograma**:*** Durante o desenvolvimento do projeto podem ocorrer desvios no cronograma. Nesta sub-atividade o gerente do projeto registra os desvios das estimativas de tempo em relação ao realizado, justificando-os.
- *Rever o Cronograma:* Durante esta sub-atividade o gerente do projeto revê o cronograma considerando a possibilidade de recuperação do desvio ou negociando com as partes envolvidas. As decisões e ações corretivas a serem tomadas só devem ser feitas após consideradas suas implicações para o projeto.
- *Comunicar Alteração no Cronograma:* Nesta sub-atividade é realizada a divulgação da alteração no cronograma às partes envolvidas no projeto.

Produto: Cronograma revisado.

Responsável: Gerente do projeto.

4.3.2 Processo de Gerência de Custos

A *gerência de custos* tem como objetivo fornecer a estimativa dos custos do projeto.

Este processo deve ser utilizado em momentos distintos: inicialmente, após serem realizadas as estimativas iniciais de tempo, são geradas as estimativas iniciais de custos ainda com uma abrangência macroscópica. Posteriormente, as estimativas realizadas são detalhadas e, se as estimativas de tempo e esforço forem revistas, as estimativas de custos também serão.

O processo de gerência de custos é composto por três atividades:

i. Estimar os custos do projeto
ii. Elaborar orçamento do projeto
iii. Controlar orçamento do projeto

i. Estimar os Custos do Projeto

Esta atividade consiste em realizar estimativas dos custos relativos a todos os recursos necessários à realização das atividades do projeto.

Sub-atividades:

- *Registrar Elementos de Custos:* Durante o planejamento do projeto, alguns recursos necessários para desenvolvê-lo são identificados. Por exemplo, no Plano do Processo de Desenvolvimento são indicados o *hardware* e *software* necessários para desenvolver o projeto e no Plano de Recursos Humanos são indicadas as pessoas necessárias ao projeto. Nesta sub-atividade os elementos de custos já identificados são registrados no Plano de Custos e o gerente do projeto identifica aqueles que ainda não foram identificados, como por exemplo energia e aluguel.
- *Atribuir valor para os Elementos de Custos:* Durante esta sub-atividade, o gerente do projeto determina valores quantitativos e financeiros (custo

unitário, quantidade e custo total) para os elementos de custos do projeto. O custo total de cada elemento identificado será o produto de seu custo unitário por sua quantidade utilizada. Para os recursos humanos e outros recursos vinculados à execução de atividades específicas do projeto, a quantidade utilizada poderá ser obtida através da análise dos dados do cronograma do projeto.

Produto: Estimativas de custos do projeto.

Responsável: Gerente do projeto.

ii. Elaborar Orçamento do Projeto

Consiste em utilizar os valores realizados na atividade anterior para elaborar o orçamento do projeto, que será o *baseline* dos custos[5] para medir o desempenho do projeto.

Durante esta atividade todos os elementos de custos e seus respectivos valores, identificados/registrados na atividade *i*, devem ser inseridos no orçamento do projeto.

Produto: Orçamento do projeto.

Responsável: Gerente do projeto.

iii. Controlar Orçamento do Projeto

Durante o desenvolvimento do projeto, o orçamento deve ser comparado, analisado e revisto sempre que necessário.

O controle de custos envolve identificar e documentar o motivo das variações, tanto positivas quanto negativas e adequar o orçamento a elas.

Sub-atividades:

- *Registrar Desvios de Orçamento:* Durante o desenvolvimento de um projeto podem ocorrer desvios no orçamento. Nesta sub-atividade o gerente do projeto registra esses desvios, justificando-os.
- *Rever Desvios de Orçamento:* Durante esta sub-atividade o gerente do projeto toma as ações corretivas ou contingenciais para os desvios

[5] *Baseline dos Custos* é o orçamento referencial que será utilizado para medir e monitorar o desempenho dos custos do projeto. É desenvolvido através da totalização das estimativas de custos por período.

identificados. O gerente do projeto calcula o valor excedente, realiza a análise e executa a solução para tratar cada desvio identificado, procurando evitar que os custos ultrapassem a disponibilidade de recursos financeiros alocada ao projeto. As decisões e ações corretivas a serem tomadas só devem ser feitas após consideradas suas implicações para o projeto.

- *Comunicar Situações de Risco:* Nesta sub-atividade o gerente comunica às partes interessadas a ocorrência de situações de risco consequentes dos desvios do orçamento.

Produto: Orçamento revisado.

Responsável: Gerente do projeto.

4.4 A Abordagem de Planejamento de Custos para a Estação TABA

A Estação TABA tem como objetivo instanciar Ambientes de Desenvolvimento de *Software* destinados a apoiar as atividades realizadas ao longo de um projeto. Para isso, deve-se, inicialmente, definir o processo de desenvolvimento a ser utilizado no projeto e, em seguida, instanciar o ambiente para apoiar a execução do processo. A abordagem proposta neste trabalho introduz o planejamento de custos à Estação TABA, mais especificamente ao ADSOrg, uma vez que, até então, não existiam ferramentas de apoio ao planejamento de custos nesses ambientes.

A abordagem de apoio ao planejamento de custos proposta por este trabalho para a Estação TABA foi dividida em duas gerências: gerência de tempo e gerência de custos do projeto. Como requisitos básicos a serem atendidos pela abordagem para a gerência de tempo tem-se: (1) apoiar a identificação das dependências entre as atividades do processo de desenvolvimento; (2) apoiar a realização das estimativas de duração das atividades do processo de desenvolvimento; (3) apoiar a elaboração do cronograma do projeto; e (4) apoiar o controle do cronograma do projeto. Como requisitos básicos a serem atendidos pela abordagem para a gerência de custos, tem-se: (1) apoiar a identificação dos elementos de custos do projeto; (2) apoiar a elaboração do orçamento do projeto; e (3) apoiar o controle do orçamento do projeto. Estes requisitos foram definidos tendo como base as

atividades que são realizadas ao longo do planejamento de custos de um projeto de *software*.

A abordagem aqui descrita está inserida no contexto de Ambientes de Desenvolvimento de *Software* Orientados à Organização e pretende, além de atender os requisitos acima, disponibilizar ao gerente de projeto todo o conhecimento de prazos e custos acumulado pelos vários gerentes da organização. Informações sobre os prazos e custos praticados em projetos similares anteriormente realizados podem ajudar o gerente do projeto a refinar suas estimativas e torná-las mais próximas dos valores reais. Além disso, as idéias, diretrizes, problemas, dúvidas e lições aprendidas relacionados à gerência de custos e registradas pelos gerentes de projetos anteriores podem apoiar no planejamento de custos de novos projetos.

O conhecimento adquirido pelos usuários de um ADSOrg instanciado, ao longo das várias atividades do processo de desenvolvimento, será armazenado no repositório do projeto. Além disso, esse repositório conterá os dados do projeto de *software* como, por exemplo, sua equipe de desenvolvimento, processo utilizado, riscos identificados, custos estimados, dentre outros. O repositório do projeto refletirá sempre a atual situação do projeto de *software* associado ao ADSOrg instanciado. Quando o projeto é concluído, os dados que representarem potenciais fontes de aprendizado organizacional serão copiados para o repositório da organização, que tem como objetivo armazenar o conhecimento obtido ao longo dos vários projetos de *software* da organização. O repositório da organização poderá ser consultado durante o desenvolvimento de um projeto de *software*, visando a utilização do conhecimento organizacional nas várias atividades do processo de desenvolvimento de *software*.

No que diz respeito ao conhecimento e dados relacionados ao planejamento de custos, o repositório do projeto armazenará os valores referentes a prazo, esforço e custos envolvidos no projeto, as idéias, sugestões e comentários que surgirem ao longo do planejamento de custos do projeto, além das lições aprendidas em gerência de tempo e custos.

No final do projeto, todo o conhecimento de prazos e custos adquirido durante o planejamento de custos será copiado do repositório do projeto para o repositório da organização, tornando-o, dessa forma, disponível para futura utilização. Além disso, os

dados de prazos e custos do projeto também serão copiados para o repositório da organização, por representarem fontes importantes de aprendizado organizacional em gerência de custos.

A seguir será descrito como as atividades relacionadas ao planejamento de custos serão realizadas em um ADSOrg. Os processos descritos na seção 4.3 foram modelados, buscando-se representar o conhecimento tácito e explícito necessários e as ferramentas utilizadas em cada atividade dos processos. As atividades dos processos de gerência de tempo e gerência de custos serão abordadas, apresentando-se sua modelagem e como a abordagem proposta pretende apoiar sua realização. Os processos foram modelados utilizando-se a abordagem de modelagem de processos organizacionais apresentada em VILLELA *et al.* (2001b), que permite a representação gráfica do conhecimento e habilidades necessários à execução das atividades do processo. A notação utilizada por esta abordagem é descrita no Anexo 3.

4.4.1 Processo de Gerência de Tempo

A abordagem proposta para o processo de gerência de tempo em um ADSOrg envolve a execução de cinco atividades: *(1)* Identificar dependências entre as atividades; (*2)* Estimar a duração das atividades com abordagem *top-down*; *(3)* Estimar a duração das atividades com abordagem *bottom-up*; *(4)* Elaborar o cronograma do projeto; e *(5)* Controlar o cronograma do projeto. Essas atividades foram descritas anteriormente, na definição do processo de gerência de tempo.

O processo de gerência de tempo é utilizado em dois momentos distintos: primeiro, para gerar as estimativas iniciais do projeto, com abrangência macroscópica e, depois, para refinar essas estimativas. Ao estar realizando as estimativas iniciais, o gerente só tem acesso à lista de atividades do processo de desenvolvimento, pois as sub-atividades só são exibidas no momento de refinamento das estimativas.

A aquisição do conhecimento durante a execução do processo é feita através da interação da ferramenta *CustPlan*, objeto deste trabalho, com uma ferramenta de Aquisição de Conhecimento (MONTONI *et al.*, 2002), que está sendo desenvolvida em outra tese de mestrado, com o objetivo de permitir o registro do conhecimento adquirido pelo usuário do ADSOrg durante as várias atividades do processo. Através dessa ferramenta será permitido

o registro de idéias, dúvidas, problemas encontrados, diretrizes e lições aprendidas pelo gerente do projeto durante a execução das atividades do processo de gerência de tempo. Após serem registradas, essas informações são, então, filtradas e empacotadas para armazenamento no repositório da organização e vinculação ao processo e atividade adequados. A partir desse momento, o novo conhecimento (agora explícito) estará acessível aos usuários da ferramenta *CustPlan*.

A figura 4.2 apresenta a modelagem do processo de gerência de tempo, detalhando suas atividades, ferramentas utilizadas e conhecimento tácito e explícito necessários. O conhecimento tácito representado na figura é a experiência em gerência de tempo do gerente do projeto. Ao longo de todas as atividades do processo, o conhecimento tácito tem o mesmo significado. O conhecimento explícito está armazenado no repositório da organização e pode ser dados de projetos anteriores ou conhecimento adquirido de gerentes de projeto da organização ao realizar as atividades deste processo.

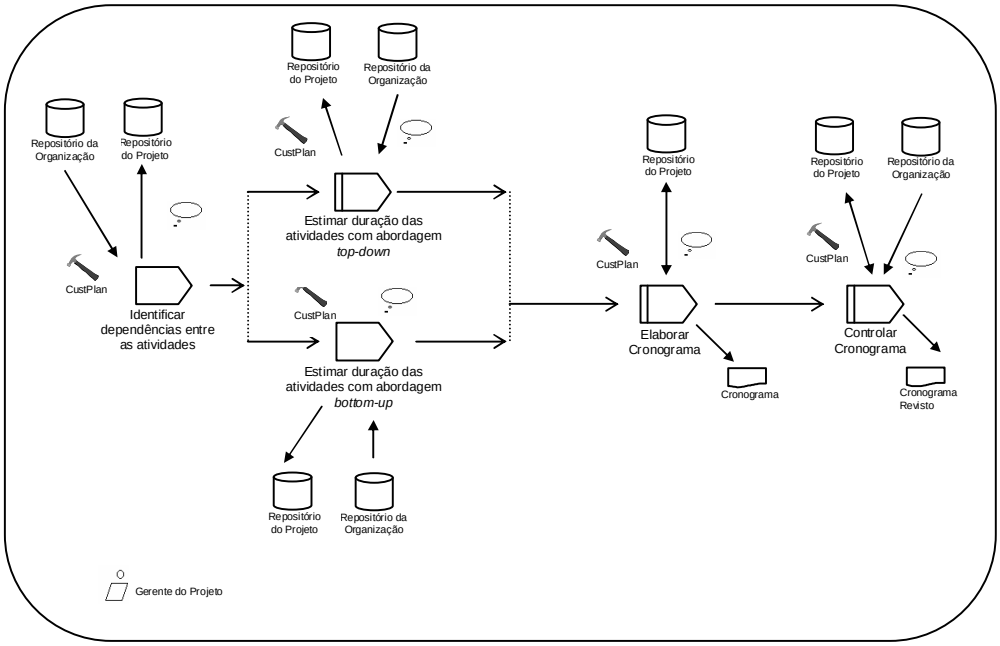


Figura 4.2 – Detalhamento do *Processo de Gerência do Tempo*

4.4.1.1 Identificar Dependências entre as Atividades do Projeto

A abordagem proposta apóia a identificação das dependências entre as atividades do processo de desenvolvimento disponibilizando dados sobre as dependências usuais[6] entre as atividades, bem como outros conhecimentos relacionados à atividade, disponíveis no repositório da organização.

Para a realização dessa atividade, o gerente deve analisar as atividades/sub-atividades do processo instanciado e, para cada uma delas, indicar quais são as atividades/sub-atividades que necessitam estar concluídas para que esta possa ser iniciada.

4.4.1.2 Estimar a duração das Atividades do Projeto com Abordagem Top-down

Nessa atividade, o gerente, inicialmente, realiza as estimativas de todo o projeto ou de seus módulos e, em seguida, distribui as estimativas entre as atividades/sub-atividades do projeto e/ou módulo.

Caso o gerente escolha estimar a duração das atividades de forma *top-down*, esta abordagem apoiará a realização das estimativas de duração das atividades do processo de desenvolvimento através de dois modelos paramétricos (Análise de Pontos de Função e COCOMO II), da disponibilização de dados sobre prazo e esforço em projetos similares anteriores e da possibilidade de consulta ao conhecimento disponível no repositório da organização relativo a essa atividade.

Para realizar as estimativas das atividades do projeto com a abordagem *top-down,* o gerente, inicialmente, deve utilizar os modelos paramétricos Análise de Pontos de Função e COCOMO II para realizar as estimativas de todo o projeto ou de seus módulos. Em seguida, o gerente do projeto deve refinar essas estimativas analisando dados de projetos similares anteriores e utilizando sua experiência pessoal para decidir as estimativas finais. Quando um modelo paramétrico é utilizado, os parâmetros e constantes a ele pertinentes são obtidos no repositório da organização, bem como os dados de projetos similares. As estimativas obtidas são, então, armazenadas no repositório do projeto. As estimativas realizadas através dos modelos paramétricos também serão armazenadas no repositório do

[6] O conhecimento disponibilizado sobre as dependência usuais é resultado de pesquisa com especialistas sobre as dependências entre as atividades do processo de desenvolvimento da ISO 12207, apresentada no capítulo 5 e descrita no Anexo 4.

projeto e com a conclusão do projeto serão copiadas para o repositório da organização. Isso possibilitará a análise e comparação com os valores realmente praticados, fornecendo assim o grau de acerto das técnicas utilizadas pela organização.

Após realizar as estimativas para todo o projeto ou para seus módulos, o gerente deve distribuí-las entre as atividades/sub-atividades envolvidas no desenvolvimento do projeto/módulos. Para isso, o gerente pode consultar dados de projetos anteriores para orientá-lo na distribuição das estimativas. A disponibilização desses dados fornecerá o prazo e esforço de uma determinada atividade em cada projeto similar encontrado e a relação desses valores com os valores totais do projeto. Por exemplo, o gerente pode observar que em um determinado projeto a atividade "Análise de Requisitos do Sistema" foi realizada em vinte dias e o projeto foi realizado em cem dias. Ou seja, a atividade em questão consumiu 20% do tempo do projeto. Com base nesses dados, o gerente realiza a distribuição das estimativas totais entre as atividades do projeto /ou seus módulos.

A figura 4.3 apresenta a modelagem da atividade *Estimar a duração das Atividades do Projeto com abordagem Top-down* , detalhando suas sub-atividades, ferramentas utilizadas e conhecimento tácito e explícito necessários.

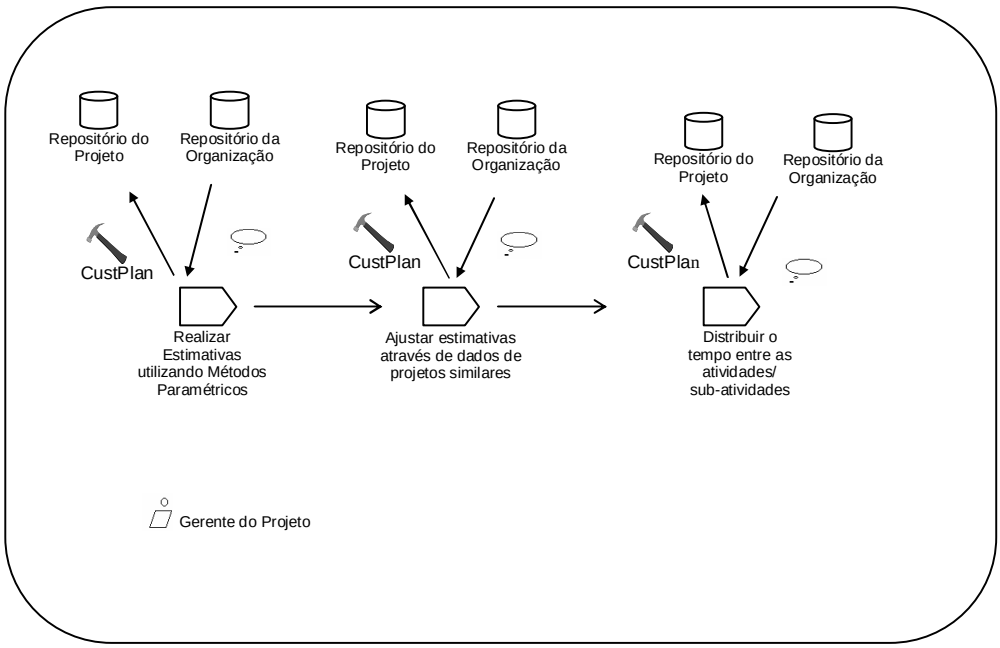


Figura 4.3 – Detalhamento da atividade *Estimar a Duração das atvidades do Projeto com Abordagem Top-down*

4.4.1.3 Estimar a duração das Atividades do Projeto com Abordagem Bottom-up

Caso seja escolhida a abordagem *bottom-up*, esta atividade apoiará a realização das estimativas de duração das atividades/sub-atividades do processo de desenvolvimento através da disponibilização de dados sobre prazo e esforço em projetos similares anteriores e através da possibilidade de consulta a outros tipos de conhecimento disponíveis no repositório da organização relativos a essa atividade.

O gerente realiza as estimativas para as atividades/sub-atividades. As estimativas totais do projeto e/ou de seus módulos são obtidas pelo somatório das estimativas para as suas atividades/sub-atividades.

4.4.1.4 Elaborar Cronograma

Para elaborar o cronograma, inicialmente, devem ser identificados os caminhos críticos do projeto. Como mencionado anteriormente, um caminho crítico é uma seqüência de atividades tais que, se houver atraso em alguma delas, esse atraso será transmitido ao término do projeto.

A identificação dos caminhos críticos será realizada, utilizando o algoritmo do *Critical Path Method - CPM* (PRADO, 1998*)*, que verifica as dependências entre as atividades e suas durações e indica quais são os caminhos críticos do projeto.

Após serem identificados os caminhos críticos, o gerente determina as datas de início e fim das atividades, observando sua duração, dependências e caminhos críticos identificados.

Um dos planos que fazem parte do Plano do Projeto é o Plano de Acompanhamento, onde são identificados os marcos e pontos de controle do projeto. O gerente deve, neste momento, registrar esses marcos e pontos de controle no cronograma do projeto.

A figura 4.4 apresenta a modelagem da atividade *Elaborar Cronograma do Projeto*, detalhando suas sub-atividades e ferramentas utilizadas.

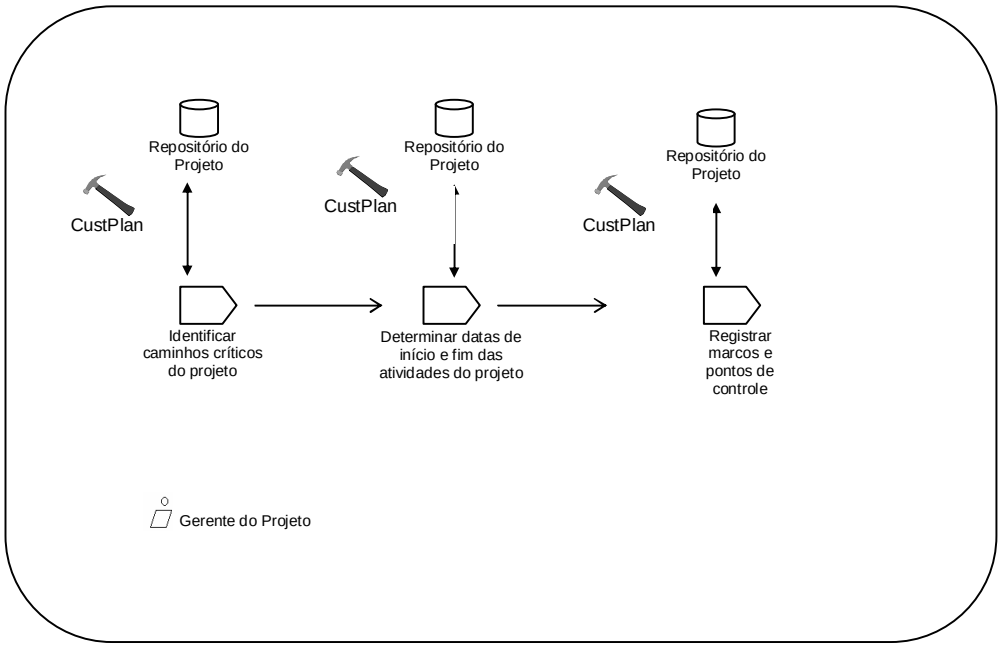


Figura 4.4 – Detalhamento da atividade *Elaborar Cronograma do Projeto*

4.4.1.5 Controlar Cronograma

Durante o desenvolvimento de um projeto podem ocorrer desvios no cronograma do projeto. O gerente de projeto deve registrar esses desvios, identificando em qual(is) atividades ocorreram e o motivo de sua ocorrência.

Após serem registrados os desvios, o gerente deve rever o cronograma considerando:

a) A possibilidade de recuperação do desvio do cronograma, revendo-o de forma a buscar alcançar o próximo marco do projeto na data prevista e garantindo o término do projeto na data esperada;
b) Caso não haja possibilidade de recuperação do atraso, o gerente deve rever o cronograma como um todo e negociar com as partes envolvidas.

Para identificar as ações corretivas a serem tomadas, o gerente do projeto deve analisar as razões dos desvios, podendo consultar o conhecimento disponível no repositório da organização. O gerente pode também analisar as outras atividades do projeto onde também ocorreram atrasos e as razões que motivaram os desvios.

Identificadas as ações corretivas e tendo sido alterado o cronograma, as mudanças devem ser comunicadas a todas as partes envolvidas.

A figura 4.5 apresenta a modelagem da atividade *Controlar Cronograma do Projeto*, detalhando suas sub-atividades, ferramentas utilizadas e conhecimento tácito e explícito necessários.

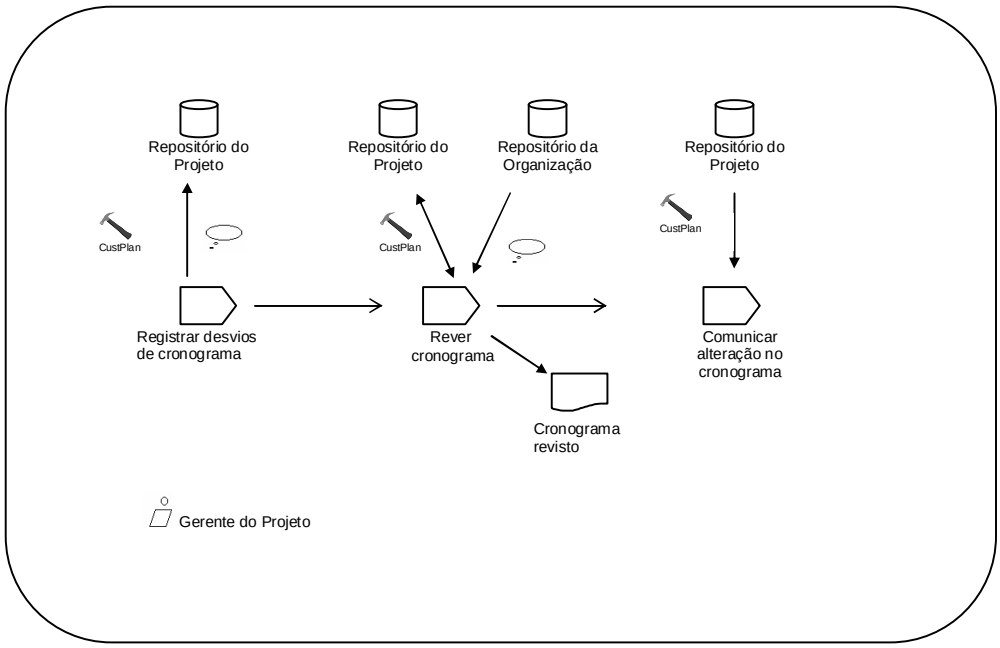


Figura 4.5 – Detalhamento da atividade *Controlar Cronograma do Projeto*

4.4.2 Processo de Gerência de Custos

A abordagem proposta para o processo de gerência de custos em um ADSOrg envolve a execução de três atividades: *(1)* Estimar custos; (*2)* Elaborar o orçamento do projeto; e *(3)* Controlar o orçamento do projeto. Essas atividades foram descritas anteriormente, na definição do processo de gerência de custos.

Assim como no processo de gerência de tempo, a aquisição do conhecimento durante a execução do processo é feita através da interação da ferramenta *CustPlan* com uma ferramenta de Aquisição de Conhecimento que permite o registro do conhecimento adquirido pelo usuário do ADSOrg durante as várias atividades do processo e armazena o conhecimento no repositório da organização. A partir desse momento, o novo conhecimento estará acessível aos usuários da ferramenta *CustPlan*.

A figura 4.6 apresenta a modelagem do processo de gerência de custos, detalhando suas atividades, ferramentas utilizadas e conhecimento tácito e explícito necessários. O conhecimento tácito representado na figura é a experiência em gerência de custos do gerente do projeto. Ao longo de todas as atividades do processo, o conhecimento tácito tem o mesmo significado. O conhecimento explícito está armazenado no repositório da organização e pode ser dados de projetos anteriores ou conhecimento adquirido de gerentes de projeto da organização ao realizar as atividades deste processo.

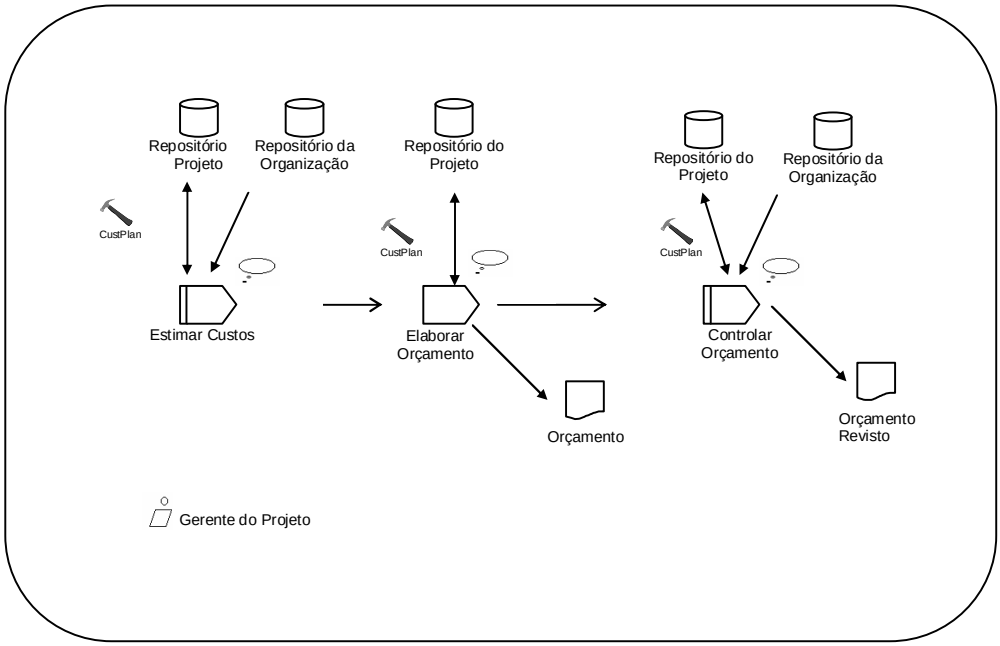


Figura 4.6 – Detalhamento do Processo de *Gerência de Custos*

4.4.2.1 Estimar Custos do Projeto

Para realizar a estimativa dos custos do projeto, o gerente, inicialmente, deve identificar os elementos de custos. Alguns desses elementos já foram identificados em outros planos do Plano do Projeto, como por exemplo no Plano de Recursos Humanos que define a equipe do projeto. Uma lista com todos os elementos de custos já identificados é fornecida ao gerente. Os elementos que não foram identificados, como por exemplo gastos com energia e aluguel , devem ser incluídos na lista pelo gerente do projeto.

Para cada elemento de custo identificado, o gerente deve determinar a quantidade que será utilizada e o custo financeiro correspondente. Para isso, o gerente deve consultar o

custo unitário de cada elemento no repositório da organização, que deverá tê-los sempre atualizado.

A figura 4.7 apresenta a modelagem da atividade *Estimar Custos do Projeto*, detalhando suas sub-atividades, ferramentas utilizadas e conhecimento tácito e explícito necessários.

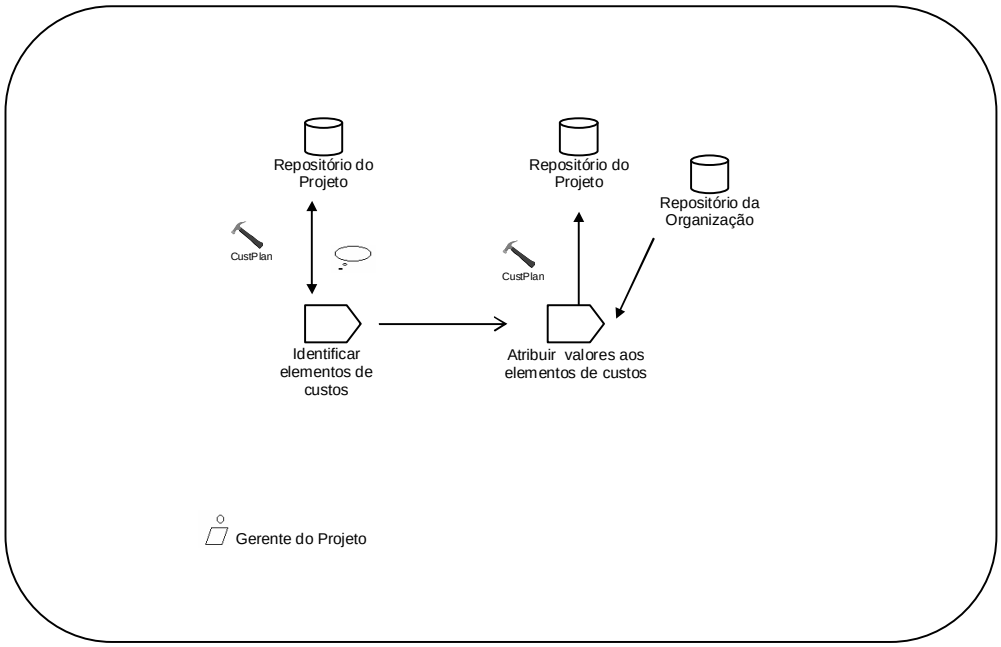


Figura 4.7 – Detalhamento da atividade *Estimar Custos do Projeto*

4.4.2.2 Elaborar Orçamento do Projeto

A elaboração do orçamento do projeto consiste na distribuição dos custos identificados na atividade anterior (4.4.2.1) ao longo do desenvolvimento do projeto, ou seja, é elaborado um cronograma financeiro que contém as receitas previstas para o projeto, as despesas e suas respectivas datas. Conforme apresentado no processo de gerência de custos, é a alocação das estimativas dos custos globais aos itens individuais de trabalho com a finalidade de estabelecer um *baseline* dos custos para medir o desempenho do projeto.

Para elaborar o orçamento, o gerente pode consultar o conhecimento disponível no repositório da organização e útil na realização dessa atividade.

4.4.2.3 Controlar Orçamento do Projeto

Durante o desenvolvimento de um projeto podem ocorrer desvios no orçamento do projeto que devem ser registrados pelo gerente.

Após o registro, o gerente deve, então, corrigir o orçamento para que o mesmo se adeque aos desvios registrados. O gerente do projeto deve analisar os desvios e tomar medidas para evitar que os custos ultrapassem a disponibilidade de recursos financeiros alocada ao projeto, podendo consultar o conhecimento disponível no repositório da organização.

Se necessário, situações graves com relação ao orçamento, que coloquem em risco o projeto, devem ser comunicadas às partes interessadas.

A figura 4.8 apresenta a modelagem da atividade *Controlar Orçamento do Projeto*, detalhando suas sub-atividades, ferramentas utilizadas e conhecimento tácito e explícito necessários.

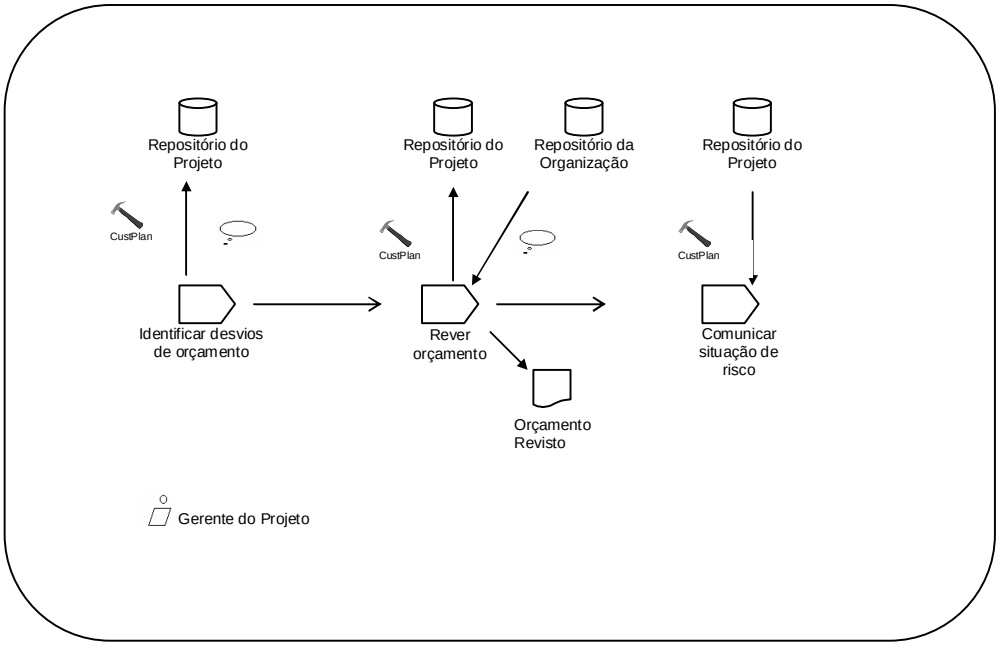


Figura 4.8 – Detalhamento da atividade *Controlar Orçamento do Projeto*

4.5 Considerações Finais

Este capítulo apresentou uma abordagem para o planejamento de custos em projetos de *software* fundamentada nos conceitos de gerência do conhecimento e Ambientes de Desenvolvimento de *Software* Orientados à Organização. A abordagem propõe a utilização, durante o planejamento de custos, do conhecimento disponível no repositório da organização referente às atividades dos processos de gerência de tempo e custos e de dados de projetos anteriores similares.

Também foram apresentados os processos de gerência de tempo e gerência de custos definidos e suas abordagens na Estação TABA.

Uma ferramenta de Aquisição de Conhecimento (MONTONI *et al.*, 2002) permitirá o registro do conhecimento obtido durante a execução dos processos de gerência de tempo e gerência de custos. Uma interface da *CustPlan* com essa ferramenta permitirá o acesso ao conhecimento disponível no repositório da organização.

Os resultados esperados pela utilização da abordagem apontam para o refinamento das estimativas, possibilitando a realização de estimativas que se aproximem o máximo possível dos valores realmente praticados, principalmente através do acesso ao conhecimento acumulado na organização e relevante para essa atividade.

Capítulo 5

A Ferramenta *CustPlan*

Neste capítulo é apresentada a ferramenta CustPlan, desenvolvida na Estação TABA para apoiar o Planejamento de Custos em ADSOrg.

5.1 Introdução

Buscando-se apoiar a abordagem de planejamento de prazos e custos em ADSOrg na Estação TABA, foi implementada a ferramenta *CustPlan*, que apóia as atividades presentes nos processos de gerência de tempo e gerência de custos apresentados no capítulo 4.

CustPlan é disponibilizada em um ADSOrg instanciado e, dessa forma, possibilita a utilização do conhecimento organizacional armazenado no repositório da organização. *CustPlan* faz parte das ferramentas disponibilizadas ao usuário do ADS durante a atividade de Planejamento do Projeto, mais especificamente durante a realização das atividades Planejamento de Tempo e Planejamento de Custos.

Este capítulo apresenta, na seção 5.2, a caracterização de projetos realizada na Estação TABA para a recuperação de projetos similares. Em seguida, na seção 5.3, é apresentada a descrição da pesquisa realizada com gerentes para determinar as dependências usuais entre as atividades do processo de desenvolvimento. Na seção 5.4 a ferramenta *CustPlan* é apresentada e na seção 5.5 são apresentadas as considerações finais.

5.2 Caracterização de Projetos na Estação TABA

A caracterização de projetos de *software* na Estação TABA tem como objetivo permitir o agrupamento de projetos que possuem características semelhantes sob um determinado aspecto. O agrupamento de projetos similares permite que sejam realizadas as analogias com os dados de projetos anteriormente realizados quando se trata de planejamento de prazos e custos. Porém, outras áreas também têm interesse em agrupar os

projetos, como por exemplo a análise de riscos, que procura por conhecimento acumulado de riscos que estão presentes em projetos similares. Sendo assim, os critérios de caracterização propostos foram separados em dois grandes grupos: critérios genéricos e critérios específicos.

Os critérios genéricos são os que podem caracterizar um projeto independente da finalidade para o qual a caracterização será utilizada. Os critérios específicos são aqueles que são definidos de acordo com a finalidade da caracterização. Por exemplo, para traçar perfis de projetos para realizar analogias para determinação de prazos e custos, um critério específico importante pode ser "*Restrição de Cronograma"*, que indica que o tempo para desenvolvimento do projeto é pré-estabelecido (como em um desenvolvimento de um site que precisa estar ativo em 3 meses).

A determinação dos critérios genéricos de um projeto é obrigatória, ou seja, esses critérios precisam ser informados para que o agrupamento de projetos similares seja realizado. Os critérios específicos devem ser informados de acordo com a finalidade da caracterização de projetos na Estação TABA.

Para utilizar os critérios de caracterização deve ser determinado um mecanismo de busca por projetos considerados similares segundo os critérios estabelecidos. Porém, essa não é uma tarefa simples.

SHEPPERD e SCHOFIELD (1997) relatam problemas encontrados na busca por projetos similares. Usando a similaridade para estimar o esforço de desenvolvimento de *software* baseado em analogia, SHEPPERD e SCHOFIELD (1997) propõem uma abordagem na qual os projetos similares são encontrados medindo-se a distância Euclidiana em um espaço n-dimensional onde cada dimensão corresponde a uma variável (um critério de caracterização). A limitação desta abordagem é que ela não trabalha com variáveis medidas em escalas nominais (exemplo: domínio de aplicação).

Como já mencionado no capítulo 2, IDRI e ABRAN (2001) consideram que a similaridade de projetos de *software* não tem sido assunto de estudos detalhados, apesar de ser frequentemente utilizada em estimativas de esforço de desenvolvimento de *software* baseadas em analogia. Eles propõem a utilização de um modelo baseado em lógica Fuzzy para definir as métricas de similaridade dos projetos de *software* e, então, superar a limitação de não poder trabalhar com atributos de projeto medidos em escalas nominais .

PRIETO-DIAZ (1991), em estudos para recuperação de componentes similares, propôs a *classificação facetada*. A proposta do autor pode ser adaptada para tornar possível a recuperação de projetos similares.

Neste trabalho não foi determinado o mecanismo de busca por projetos similares, dada a complexidade de tal tarefa. Inicialmente, a busca será realizada através da igualdade de todos os critérios avaliados. Futuramente, uma técnica de busca por projetos similares deverá ser desenvolvida.

As tabelas a seguir apresentam os critérios de caracterização genéricos e específicos propostos para a Estação TABA. Os critérios foram escolhidos com base na literatura, principalmente nas propostas de JONES (2000), e levando-se em consideração os critérios que já estavam presentes na Estação TABA.

Critério	**Descrição**
Indústria	Define a indústria na qual o software está inserido. Pode-se usar a classificação padrão de indústrias para determinar o conjunto de indústrias existentes. Exemplos são Extração de óleo e gás; Indústrias de refinaria de petróleo e relacionadas; Comunicação, Serviços de Transporte (JONES, 2000).
Tipo de Software	Define a área de aplicação à qual o software se destina. Exemplos de tipos de software são: Sistemas Especialistas, Sistemas de Informação, Softwares Embutidos, Sistemas Militares, etc.
Paradigma	Indica o paradigma utilizado no desenvolvimento do projeto. Ex.: Estrutural, Orientação a Objetos, etc.
Natureza do Projeto	Indica se o projeto é um projeto de desenvolvimento ou se envolve alguma forma de manutenção. As possíveis naturezas de projeto são: novo desenvolvimento, melhoria (adição de funções), atualização para atender a novas regulamentações, reparo de defeitos, melhoria de performance, migração para nova plataforma, nacionalização, reengenharia, atualização em massa (por exemplo: Euro e ano 2000), híbrida (JONES, 2000).

Tabela 5.1 – Critérios Genéricos de Caracterização de Projetos de *Software*

Critério	**Descrição**
Nível de experiência dos gerentes do projeto	Indica o nível de experiência da gerência do projeto com Engenharia de Software, com a plataforma de desenvolvimento, a tecnologia utilizada e por último com o domínio da aplicação (JONES, 2000; MENZIES e SINSEL, 2000; IDRI e ABRAN, 2001). Para diminuir a subjetividade deste critério são utilizados 5 valores possíveis para cada tópico de experiência: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (GOMES, 2001).
Nível de experiência da equipe de desenvolvimento	Indica o nível de experiência da equipe de desenvolvimento com Engenharia de Software, com a plataforma de desenvolvimento, a tecnologia utilizada e por último com o domínio da aplicação (JONES, 2000; MENZIES e SINSEL, 2000; IDRI e ABRAN, 2001). Para diminuir a subjetividade deste critério são utilizados 5 valores possíveis para cada tópico de experiência: 0 – nenhuma experiência; 1 – treinamento acadêmico; 2 – prática em até três projetos; 3 – experiente; 4 – capaz de orientar outros (GOMES, 2001).
Nível de experiência dos clientes	Indica o nível de experiência do cliente com ciclo de vida de desenvolvimento de software e com o domínio da aplicação (JONES, 2000).
Distribuição geográfica da equipe	Indica se a equipe está centralizada ou dispersa geograficamente (JONES,2000).
Restrição de Cronograma	Indica se o projeto possui alguma restrição de cronograma
Restrição de Desempenho ou Tempo de Execução	Indica se o projeto possui alguma restrição de desempenho ou tempo de execução (JONES, 2000; MENZIES e SINSEL, 2000; IDRI e ABRAN, 2001).
Restrição de Segurança	Indica se o projeto possui alguma restrição de segurança (JONES, 2000; MENZIES e SINSEL, 2000; IDRI e ABRAN, 2001).
Restrição de Recursos Humanos	Indica se o projeto possui alguma restrição de recursos humanos.
Uso de Tecnologia inovadora	Indica se o projeto possui algum grau de inovação relacionada a plataforma utilizada, linguagem de programação utilizada, arquitetura do projeto, etc.

Tabela 5.2 – Critérios Específicos de Caracterização de Projetos de *Software*

O projeto de *software* é caracterizado, inicialmente, no meta ambiente, no momento da definição do projeto. Antes que um ADS seja instanciado, ele é caracterizado e tem seu processo de desenvolvimento definido. A figura 5.1 ilustra a tela de Registro do Projeto, onde as informações básicas do projeto são fornecidas. A figura 5.2 exibe a tela onde as características gerais são preenchidas.

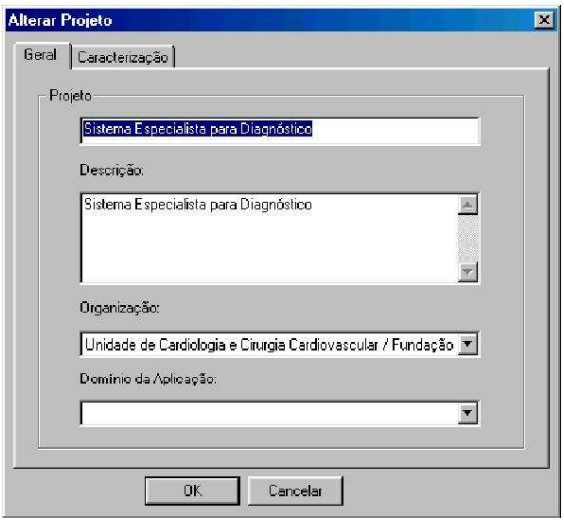


Figura 5.1 – Registro de um projeto de *software* na Estação TABA.

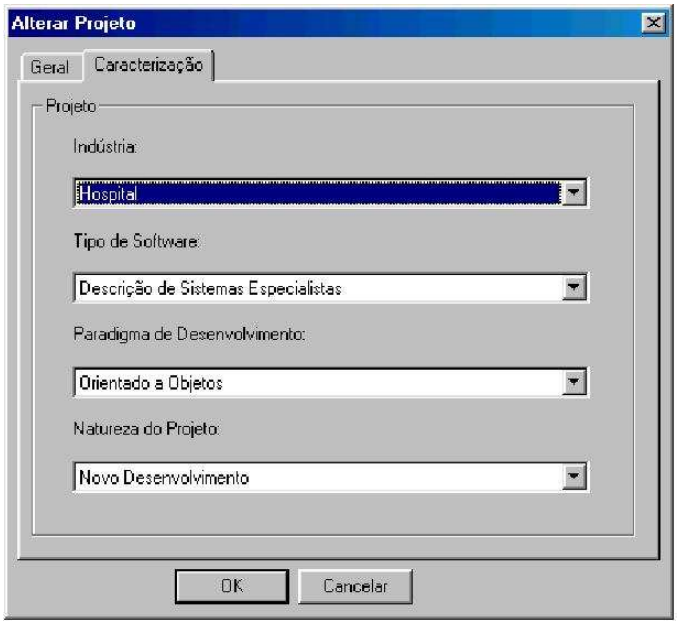


Figura 5.2 – Caracterização de um projeto de *software* na Estação TABA.

Após o projeto ter sido registrado e caracterizado de forma geral, seu processo de desenvolvimento é definido e instanciado. A figura 5.3 ilustra a tela de um ambiente instanciado, destacando a atividade de Planejamento do Projeto.

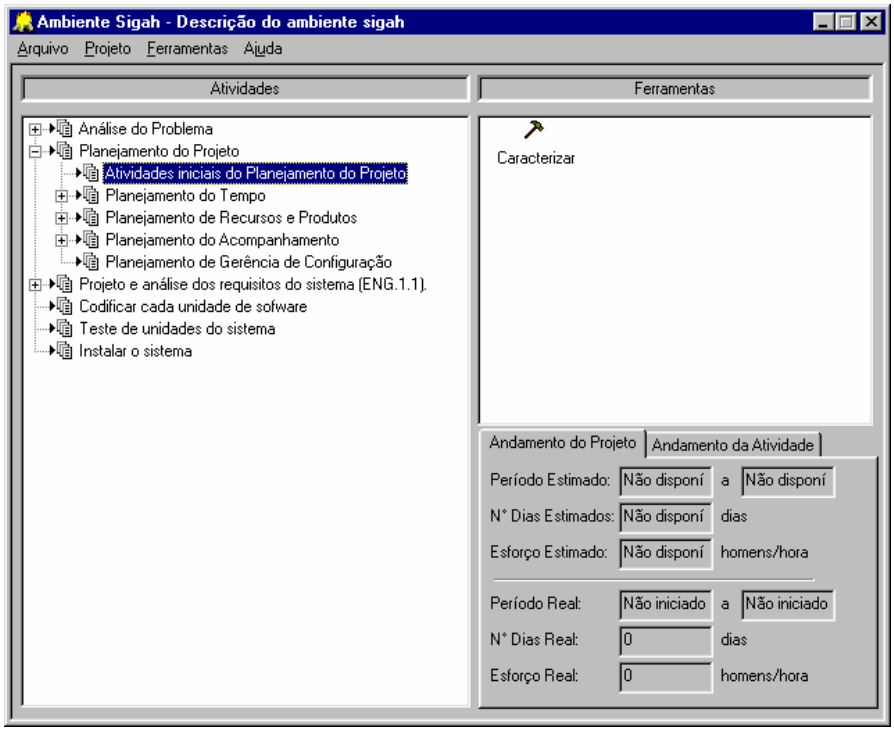


Figura 5.3 – Ambiente Instanciado pela Estação TABA.

Como pode ser visto na figura, a tela do ambiente instanciado na Estação TABA contém as atividades do processo de desenvolvimento e as ferramentas que podem ser utilizadas para a realização de cada uma dessas atividades. Essa interface proporciona ao engenheiro de *software* uma visão geral de todas as atividades associadas ao processo, bem como a relação de hierarquia e ordem de execução entre elas.

Na figura 5.3 encontra-se selecionada a atividade "Atividades Iniciais do Planejamento do Projeto". A atividade permite a caracterização específica do projeto de *software*. A ferramenta *Caracterizar* é uma ferramenta que guia o engenheiro de *software* durante o preenchimento dos critérios específicos de caracterização do projeto.

A figura 5.4 mostra uma das telas da ferramenta *Caracterizar* onde os critérios relacionados à equipe do projeto são preenchidos.

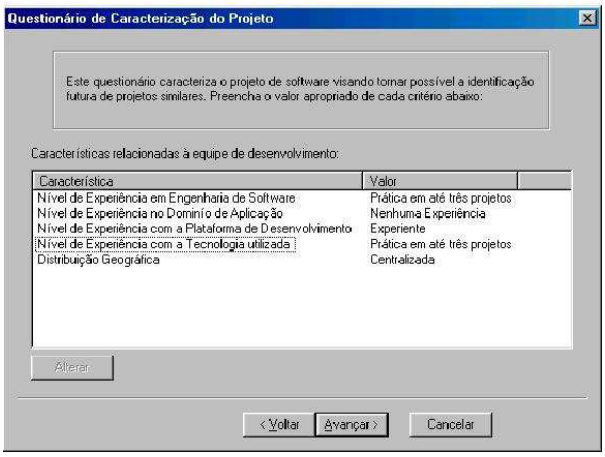


Figura 5.4 – Caracterização específica de um projeto de *software* na Estação TABA.

5.3 Pesquisa de Dependências Usuais entre as Atividades do Processo de Desenvolvimento

O primeiro passo para realizar o planejamento do cronograma do projeto é indicar as dependências entre as atividades do projeto, conforme apresentado no processo de gerência de tempo, descrito no capítulo 4. Para realizar esse passo utilizando *CustPlan*, o gerente de projetos poderá consultar no repositório da organização as dependências usuais entre as atividades do processo de desenvolvimento.

Para coletar as dependências usuais, foi realizada uma pesquisa cujos participantes foram gerentes de projetos de *software* da Fundação COPPETEC (Coordenação de Projetos, Pesquisas e Estudos Tecnológicos) e outros gerentes de projetos que atuam em processos de desenvolvimento do *software*. Foram considerados 8 gerentes de projeto, sendo estes mestres ou doutores na área de informática e com experiência relevante em gerência de projetos de *software*.

A pesquisa foi realizada utilizando-se um questionário que considerou as atividades do processo de desenvolvimento da ISO/IEC 12207 - Tecnologia de Informação – Processos de ciclo de vida de *software*. O questionário apresentou uma tabela onde o participante indicou, para cada atividade do processo, suas pré-atividades, ou seja, as atividades que precisam estar concluídas para que ela possa ser realizada. Foi permitido ao participante registrar considerações sobre as dependências e comentários em geral.

Com base nos dados coletados, um conjunto de dependências usuais entre as atividades do processo de desenvolvimento da ISO/IEC 12207 foi proposto e armazenado no repositório da organização para ser utilizado durante a atividade *Identificar as Dependências entre as Atividades do Projeto* realizada no planejamento do cronograma do projeto.

Para a realização dessa pesquisa foram seguidas cinco atividades: *(i)* definição; *(ii)* planejamento; *(iii)* operação; *(iv)* análise e interperetação; e *(v)* apresentação. O ponto de partida foi a concepção da idéia da pesquisa, onde foi avaliado se uma pesquisa era realmente necessária e apropriada para atender as questões a serem investigadas. Dando início ao processo de realização da pesquisa propriamente dito, na atividade *definição* a pesquisa foi definida em termos do problema pesquisado, seus objetivos e metas. Na atividade de *planejamento*, o projeto da pesquisa foi realizado, a instrumentação foi elaborada e as ameaças à perfeita execução da pesquisa foram avaliadas. Durante esta atividade foi gerado o documento de planejamento da pesquisa, descrito no Anexo 4. Na atividade *operação*, os questionários foram entregues aos participantes e os dados obtidos foram organizados em uma planilha visando facilitar a análise e avaliação dos dados realizada na atividade posterior de *análise e interpretação*. Finalmente os resultados foram apresentados na atividade *apresentação*.

5.3.1 Resultados da Pesquisa

Os resultados da pesquisa caracterizaram um conjunto inicial de dependências usuais entre as atividades do processo de desenvolvimento da NBR ISO/IEC 12207 que foi armazenado no repositório da organização para apoiar os gerentes de projeto na identificação das dependências entre as atividades do projeto.

Para realizar a pesquisa, um conjunto inicial de dependências foi proposto, porém, os participantes da pesquisa não tiveram acesso às dependências presentes nesse conjunto.

Para obter os resultados, inicialmente, as características dos participantes da pesquisa foram analisadas e, em seguida, foram estabelecidos pesos para cada um deles, considerando o tempo de atuação em gerência de projetos, o número de projetos gerenciados, a experiência em processos de *software* e o conhecimento sobre a norma

NBR ISO/IEC 12207 de cada participante. A análise das características e cálculo dos pesos dos participantes os classificou em dois grupos: *experientes* e *pouco experientes*.

Foi, então, estabelecido o valor do ponto de inclusão, para indicar o valor a partir do qual uma dependência identificada pelos participantes faria parte do conjunto final de dependências.

Analisadas as dependências identificadas por cada participante, foi possível perceber que os participantes pertencentes ao grupo *experiente* identificaram as dependências presentes no conjunto inicial proposto, mesmo sem terem tido acesso a ele. Por outro lado, alguns participantes do grupo *pouco experiente* identificaram muitas dependências incoerentes.

Utilizando os pesos dos participantes e o critério de inclusão de uma dependência no conjunto de dependências (valor do ponto de inclusão), o conjunto de dependências obtido foi igual ao conjunto inicialmente proposto.

A tabela 5.3 apresenta o conjunto de dependências usuais obtido como resultado da pesquisa.

A descrição detalhada da pesquisa contendo seu objetivo, apresentação do questionário e resultados obtidos é realizada no Anexo 4.

Atividades do Processo de Desenvolvimento da NBR ISO/IEC 12207 -- Tecnologia de Informação – Processos de ciclo de vida de software que devem ser executadas	Atividades que precisam estar concluídas												
	Análise dos Requisitos do Sistema	Análise dos Requisitos do Software	Apoio à aceitação do Software	Codificação e Testes do Software	Implementação do processo	Instalação do Software	Integração do Sistema	Integração do Software	Projeto da Arquitetura do Sistema	Projeto da Arquitetura do Software	Projeto Detalhado do Software	Teste de Qualificação do Sistema	Teste de Qualificação do Software
Análise dos Requisitos do Sistema: descrição das funções, capacidades, requisitos e restrições do sistema.					X								
Análise dos Requisitos do Software: descrição das funções, capacidades, requisitos e restrições de cada item de software do sistema.	X				X								
Apoio à aceitação do Software: acompanhamento à utilização do software.	X	X		X	X	X	X	X	X	X	X	X	X
Codificação e Testes do Software: implementação e testes do software e bases de dados.	X	X			X				X	X	X		
Implementação do processo : Definição do modelo de ciclo de vida, atividades e tarefas do projeto.													
Instalação do Software: implantação do software no ambiente do cliente.	X	X		X	X		X	X	X	X	X	X	X
Integração do Sistema: integração dos itens de software ao sistema.	X	X		X	X			X	X	X	X		X
Integração do Software: integração dos componentes que compõem cada item de software.	X	X		X	X				X	X	X		
Projeto da Arquitetura do Sistema: definição dos itens de hardware, software e operações do sistema.	X				X								
Projeto da Arquitetura do Software: definição dos itens de hardware, software e operações do sistema de cada item de software do sistema.	X	X			x				X				
Projeto Detalhado do Software: refinamento do projeto da arquitetura de cada componente de software, interface e bases de dados.	X	X			X				X	x			
Teste de Qualificação do Sistema: realização de testes segundo os requisitos de qualidade estabelecidos para o sistema.	X	X		X	X		X	X	X	X	X		X
Teste de Qualificação do Software: realização de testes segundo os requisitos de qualidade estabelecidos para cada item de software.	X	X		X	X			X	X	X	X		

Tabela 5.3 – Conjunto final de dependências usuais entre as atividades do processo de desenvolvimento.

5.4 A Ferramenta *CustPlan*

Como mencionado anteriormente, a ferramenta *CustPlan* foi desenvolvida para apoiar o planejamento de custos nos Ambientes de Desenvolvimento de *Software* Orientados à Organização. Para prover esse apoio, a ferramenta é utilizada para planejar o tempo e os custos propriamente ditos, sendo acessada em diversos momentos durante a execução do processo de desenvolvimento.

O acesso à *CustPlan* é feito através da tela principal de um ADSOrg instanciado. Essa tela contém as atividades do processo de desenvolvimento e as ferramentas que podem ser utilizadas para a realização de cada uma dessas atividades. Essa interface proporciona ao engenheiro de *software* uma visão geral de todas as atividades associadas ao processo, assim como a relação de hierarquia e ordem de execução entre elas.

Para apoiar os processo de gerência de tempo e gerência de custos, *CustPlan* foi dividida em duas partes: tempo e custos. Inicialmente, o gerente utiliza *CustPlan* para realizar o planejamento do cronograma do projeto, ainda com uma visão macroscópica, considerando apenas as atividades do processo de desenvolvimento. Após o cronograma ser elaborado, o gerente realiza a alocação dos recursos humanos às atividades do projeto, utilizando a ferramenta *RHPlan* (SCHNAIDER, 2003). Em seguida, o gerente volta a utilizar a *CustPlan,* agora para elaborar o orçamento do projeto, baseando-se no cronograma e alocação de recursos realizada. Após a especificação de requisitos do sistema estar concluída, o gerente utiliza *CustPlan* para refinar o cronograma e o orçamento do projeto, considerando também as sub-atividades do processo de desenvolvimento e as informações obtidas durante a elaboração da especificação de requisitos.

5.4.1 Planejamento do Tempo

O primeiro acesso à *CustPlan* é realizado para apoiar a realização das primeiras estimativas de tempo do projeto. É feito quando o gerente seleciona, no ADSOrg instanciado, a sub-atividade “Planejamento Inicial do Tempo”, presente na atividade “Planejamento do Projeto”. Nesse momento, um ícone da *CustPlan* fica disponível no lado direito da tela.

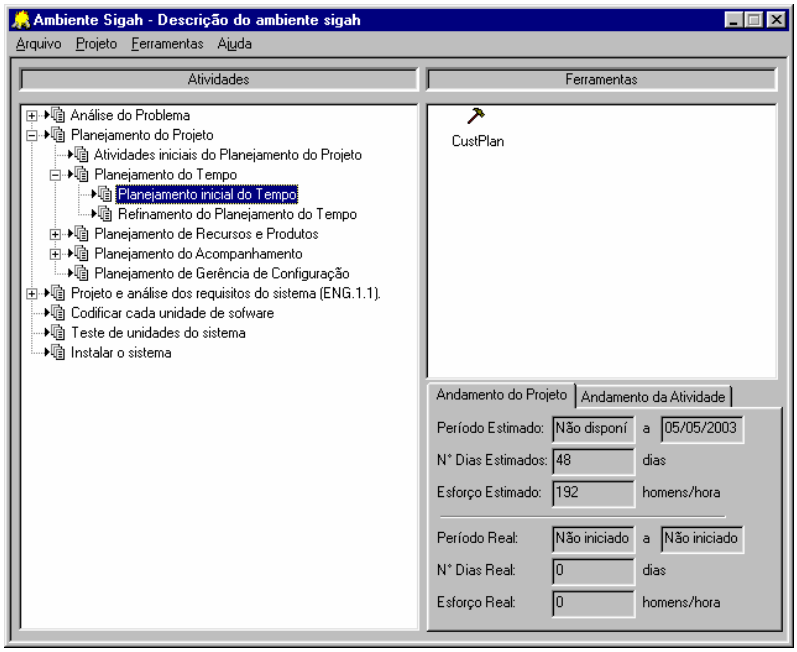


Figura 5.5 – Tela principal de um ADSOrg instanciado com acesso à CustPlan para Planejamento do Tempo

Para realizar o planejamento do tempo, a *CustPlan* apóia as seguintes atividades: *(i)* Identificar dependências entre as atividades do projeto; *(ii)* Estimar a duração das atividades do projeto*; (iii)* Elaborar o cronograma do projeto; e, *(iv)* Controlar o cronograma do projeto. Durante o planejamento inicial do tempo, como já mencionado, apenas as atividades do processo de desenvolvimento são consideradas. As sub-atividades serão consideradas no momento de refinar as estimativas geradas durante o planejamento inicial.

5.4.1.1 Identificar Dependências entre as Atividades do Projeto

Para identificar as dependências entre as atividades do projeto, o gerente realiza a atividade *Identificar Dependências* situada do lado esquerdo na tela da *CustPlan*. O gerente identifica as dependências de cada atividade do processo de desenvolvimento (exibidas no quadro "Atividades") marcando no *checklist* "Pré-Atividades" as atividades do processo de desenvolvimento que devem estar concluídas para que a atividade selecionada no quadro "Atividades" possa ser executada. O gerente pode consultar uma lista de dependências usuais entre as atividades do processo de desenvolvimento da ISO 12207, disponibilizada

no quadro "Conhecimento de Especialistas", para apoiá-lo na identificação das dependências.

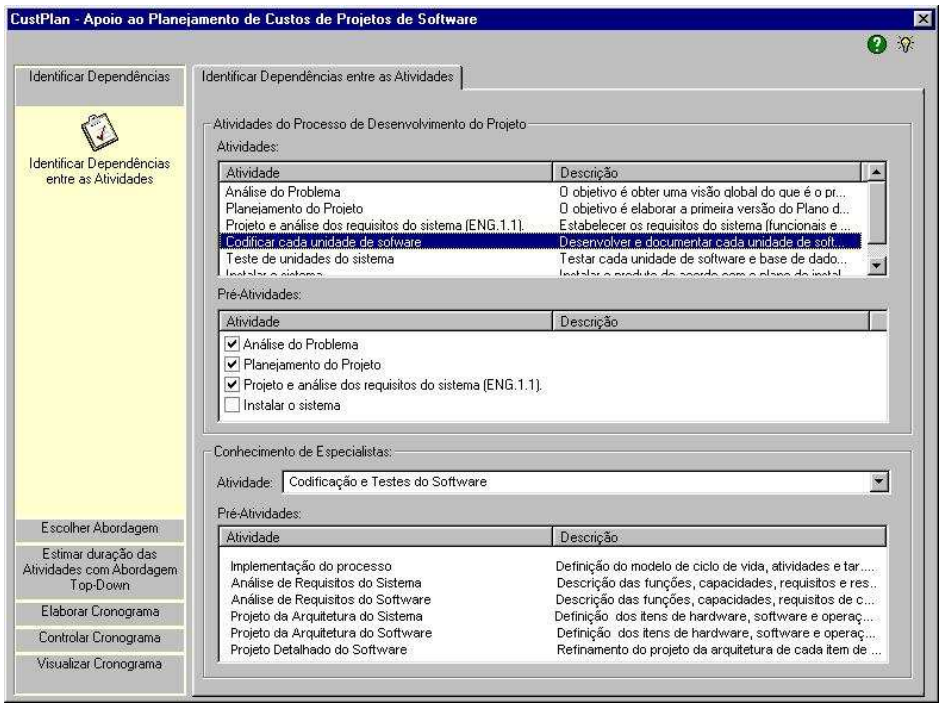


Figura 5.6 – Tela de Identificação das dependências entre as atividades do projeto

Note que a interface da *CustPlan* provê a visualização do processo que está sendo realizado (tempo ou custos) e suas atividades. No lado esquerdo é possível identificar o processo e no lado direito da interface identifica-se a atividade que está sendo realizada pelo gerente. Os ícones localizados no canto superior direito da tela permitem a realização de consulta e registro de conhecimento pertinentes às atividades do processo. O gerente do projeto pode consultar o conhecimento registrado por outros gerentes de projetos e registrar o conhecimento adquirido por ele durante a execução da atividade.

Ao longo dos processos de gerência de tempo e custos, *CustPlan* disponibiliza o conhecimento explícito armazenado para a atividade que está sendo realizada pelo gerente. A figura 5.7 apresenta a tela de consulta ao conhecimento armazenado no repositório da organização, ilustrando a consulta ao conhecimento armazenado para a atividade *Identificar dependências entre as atividades do projeto*, do processo de gerência de tempo. Esta tela é acessada através do ícone situado na parte superior da tela, mais à direita (lâmpada).

Figura 5.7 – Tela de consulta ao conhecimento armazenado no repositório da organização

O ícone em forma de interrogação situado no canto superior direito da tela da *CustPlan* realiza a interface com a ferramenta de Aquisição de Conhecimento, onde o gerente pode registrar o conhecimento adquirido durante a execução das atividades dos processos.

5.4.1.2 Estimar Duração das Atividades do Projeto

Para estimar a duração das atividades do projeto, o gerente deve, inicialmente, selecionar a atividade *Escolher Abordagem* na ferramenta. Nessa tela, o gerente indica qual será a abordagem utilizada para estimar a duração das atividades do projeto: abordagem *top-down* ou abordagem *bottom-up*. É importante ressaltar que o gerente poderá utilizar apenas uma das abordagens.

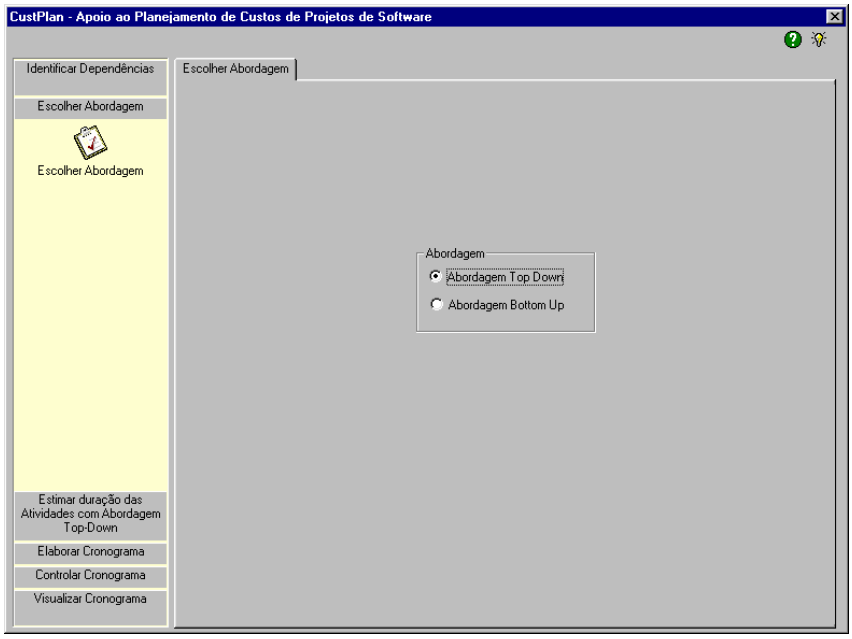


Figura 5.8 – Tela de Escolha da abordagem para estimar a duração das atividades do projeto

Após escolhida a abordagem, o gerente realiza as estimativas de acordo com a abordagem por ele determinada.

(a) Suponhamos que o gerente tenha escolhido a abordagem *Top-down.*

Conforme mencionado no capítulo 2, a associação de tipos de modelos diferentes para realizar as estimativas de projetos tem-se mostrado o caminho mais eficiente. Baseando-se nesses resultados, *CustPlan* permite, através da abordagem *top-down* de realização de estimativas, que o gerente realize as estimativas utilizando modelos paramétricos, analogia de estimativas e que utilize o conhecimento organizacional e sua experiência pessoal para decidir os valores das estimativas do projeto.

Para realizar as estimativas segundo essa abordagem, o gerente, inicialmente, seleciona a sub-atividade *Realizar as estimativas do projeto utilizando Análise de Pontos de Função*. Na tela em questão, o gerente escolhe o critério de realização das estimativas: para todo o projeto ou para um módulo específico. Se for para um módulo do projeto o gerente indica qual módulo será analisado. Em seguida, o gerente determina o número de arquivos lógicos internos e de arquivos de interface externa do sistema e, para cada um deles, o número de tipos de elementos de dados e de tipos de elementos de registros. O gerente indica, também, o número de entradas, saídas e consultas externas do sistema e, para cada uma delas, o número de arquivos referenciados e de elementos de dados envolvidos.

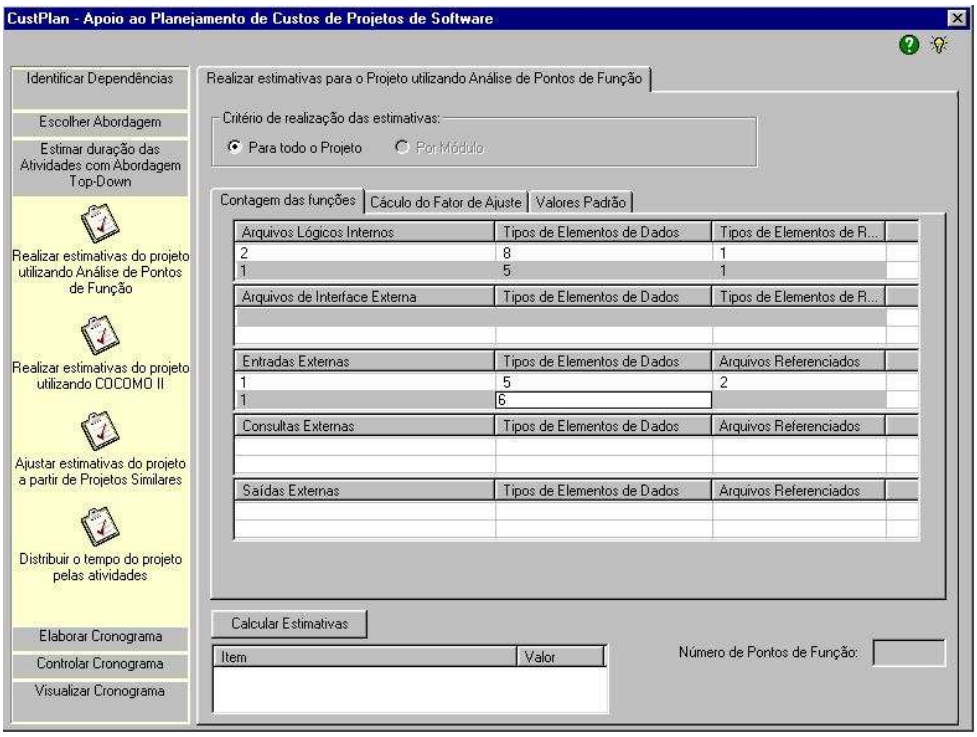


Figura 5.9 – Tela de Realização das estimativas do projeto utilizando Análise de Pontos de Função – Contagem das Funções

Após a contagem das funções, o gerente fornece o nível de influência de 14 características para o projeto, para que seja realizado o cálculo do fator de ajuste.

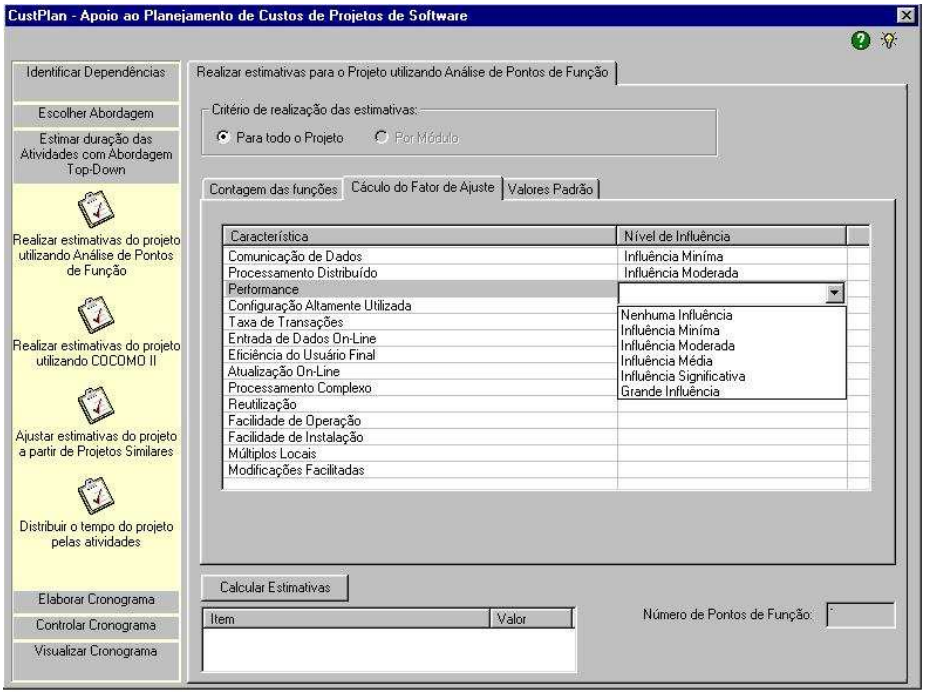


Figura 5.10 – Tela de Realização das estimativas do projeto utilizando Análise de Pontos de Função – Cálculo do Fator de Ajuste

O gerente, então, fornece valores de referência (tempo e esforço de um ponto de função na organização) para a realização do cálculo das estimativas. Fornecidos todos os valores de entrada, o gerente clica no botão "Calcular Estimativas" e estas são calculadas.

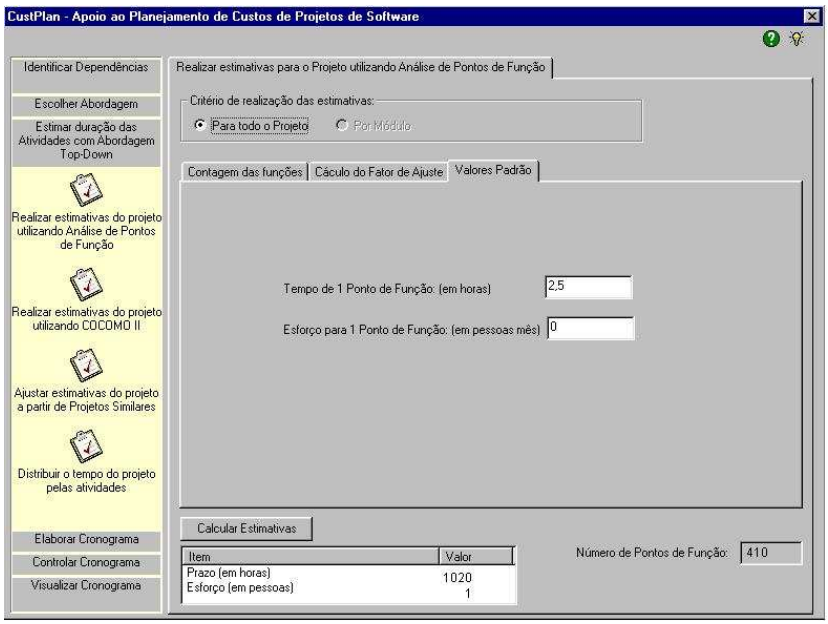


Figura 5.11 – Tela de Realização das Estimativas do projeto utilizando Análise de Pontos de Funcão – Cálculo das Estimativas

Caso o gerente não informe o esforço de referência para um ponto de função, a técnica considera o esforço de uma pessoa para o projeto. O gerente poderá ajustar essa estimativa posteriormente.

Após utilizar a técnica Análise de Pontos de Função para realizar as estimativas, o gerente seleciona a atividade *Realizar Estimativas utilizando COCOMO II.*

Para realizar as estimativas utilizando o COCOMO II, o gerente escolhe o modelo que será utilizado (Pré-Projeto ou Pós-Arquiterura). O critério de realização das estimativas para todo o projeto ou por módulos permanece o mesmo que foi determinado na Análise de Pontos de Função. O gerente indica, também, a linguagem de programação que será utilizada no projeto. Em seguida, determina os níveis de influência dos fatores de equilíbrio e direcionadores de custos para o projeto em questão.

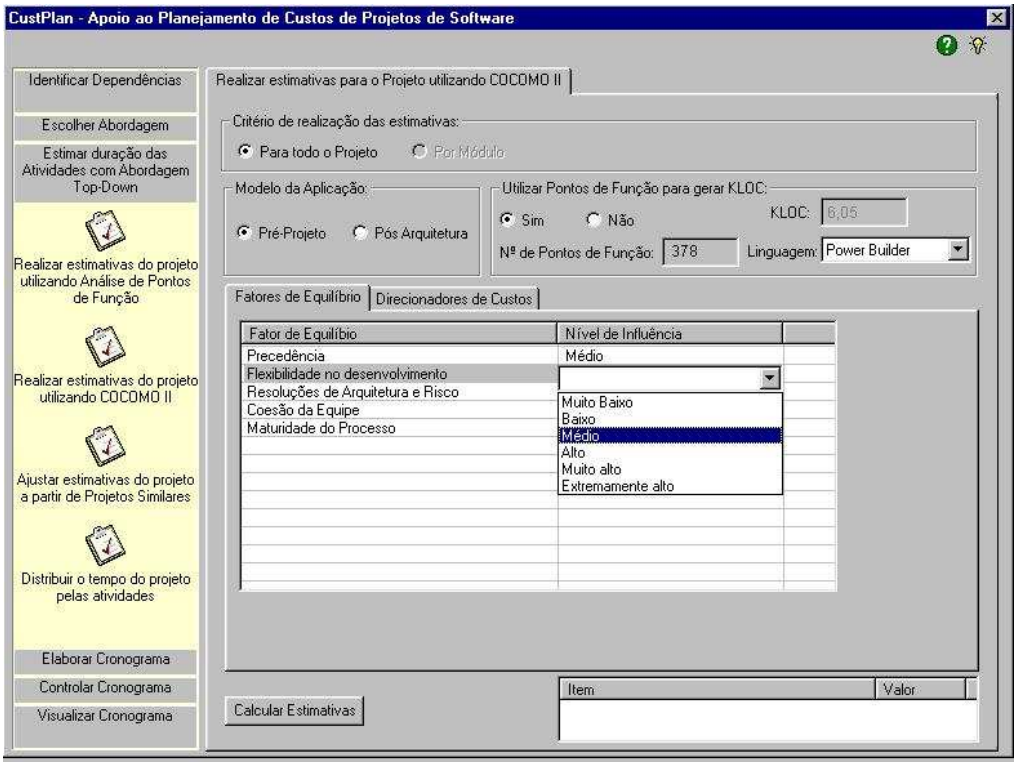


Figura 5.12 – Tela de Realização das estimativas do projeto utilizando COCOMO II – Fatores de Equilíbrio

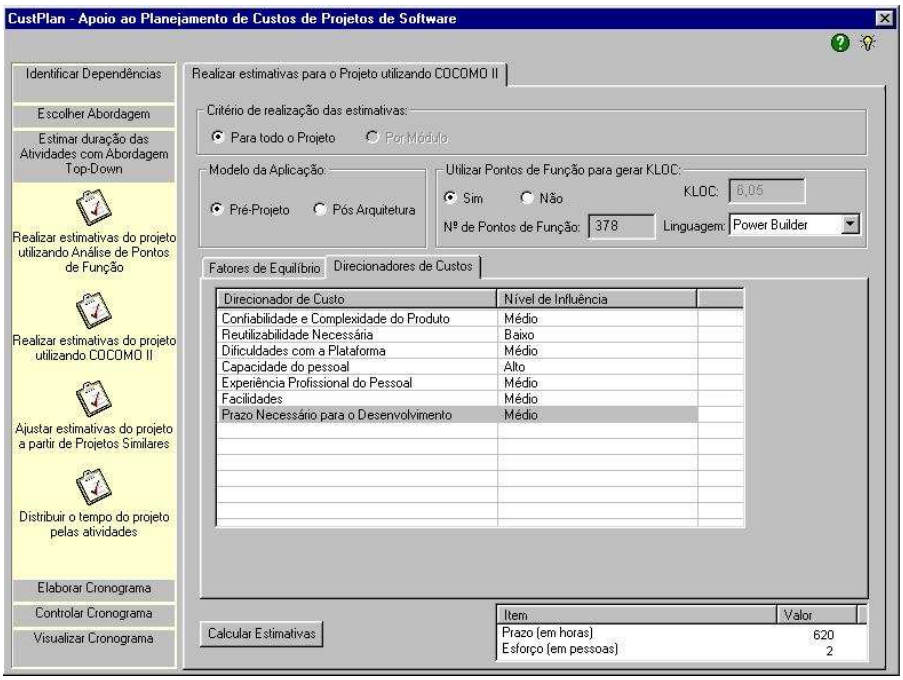


Figura 5.13 – Tela de Realização das estimativas do projeto utilizando COCOMO II – Direcionadores de Custos e Cálculo das Estimativas

Após utilizar o COCOMO II, o gerente analisa os valores das estimativas obtidas por essa técnica e pela Análise de Pontos de Função, realiza a analogia de estimativas e utiliza sua experiência pessoal para decidir as estimativas do projeto.

Para isso, ele deve selecionar a sub-atividade *Ajustar Estimativas a partir de Projetos Similares*. Nesta sub-atividade, o gerente compara as estimativas geradas pelas técnicas utilizadas e visualiza dados de projetos anteriores que irão auxiliá-lo na decisão dos valores das estimativas do projeto.

Para selecionar os projetos similares, o gerente indica no *checklist* "Critérios de Caracterização do Projeto" os critérios que devem ser utilizados na seleção dos projetos similares. Em seguida, o gerente escolhe a opção de filtro da pesquisa e clica no botão "Pesquisar". Os projetos similares são exibidos juntamente com os valores de prazo e esforço para eles realizados.

Analisando os dados dos projetos similares e os valores das estimativas do projeto geradas pelas técnicas paramétricas APF e COCOMO II, o gerente utiliza sua experiência e

determina os valores que serão efetivamente estimados para o projeto registrando-os no quadro “Valores das Estimativas”.

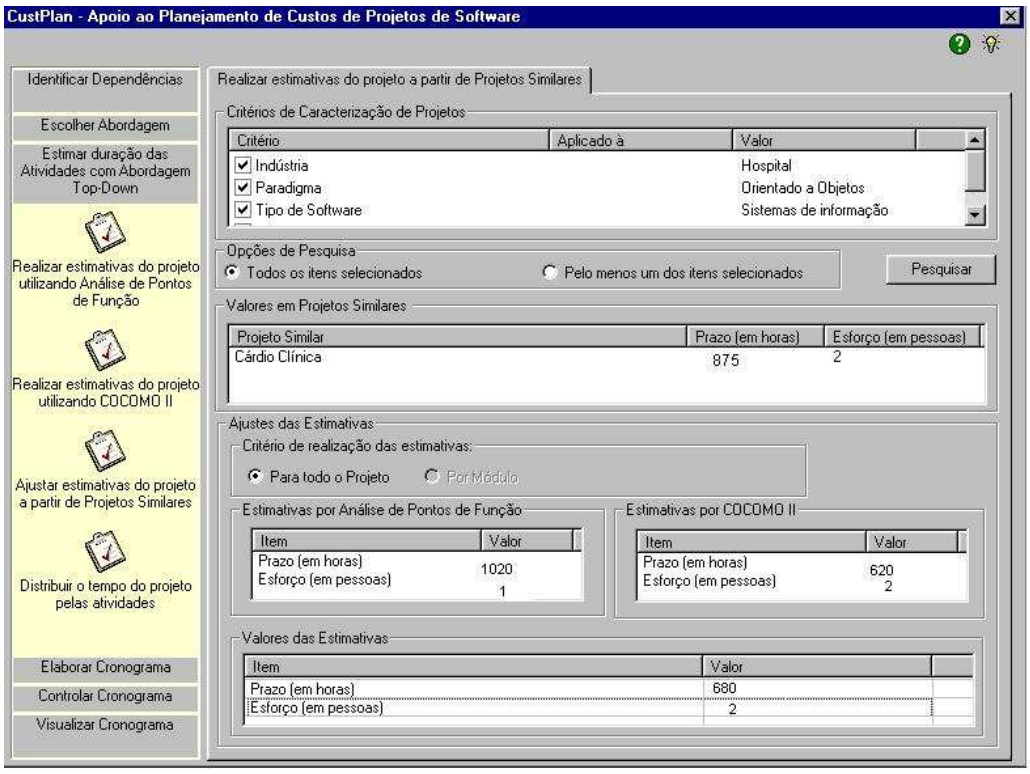


Figura 5.14 – Tela de Ajuste das estimativas a partir de dados de projetos similares

Decididos os valores das estimativas de prazo e esforço, estes devem ser distribuídos entre as atividades do processo de desenvolvimento do projeto. Para realizar as estimativas das atividades, o gerente seleciona a sub-atividade *Distribuir o tempo do projeto entre as atividades.* Nessa tela são exibidas as estimativas totais do projeto e as atividades do processo de desenvolvimento. O gerente, então, indica o prazo e esforço para cada uma das atividades, de modo que o somatório de suas estimativas seja equivalente às estimativas totais propostas.

Para auxiliar o gerente na distribuição das estimativas do projeto entre as atividades, são disponibilizados dados de projetos similares. Para cada atividade que o gerente analisar, são exibidos seu prazo e esforço nos projetos similares, bem como o percentual desses valores em relação aos valores totais do projeto.

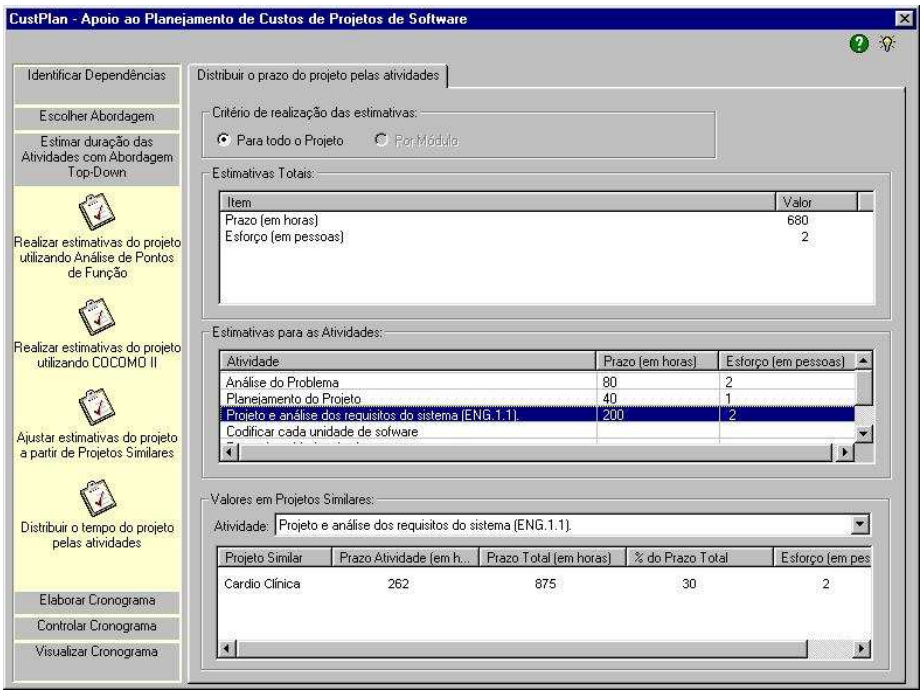


Figura 5.15 – Tela de Distribuição das estimativas entre as atividades do projeto

(b) Suponhamos, agora, que o gerente tenha escolhido a abordagem *Bottom-up*.

Para realizar as estimativas do projeto utilizando a abordagem *bottom-up*, o gerente realiza a busca por projetos similares, selecionando os critérios de caracterização e o filtro da consulta e, em seguida, analisa os dados dos projetos similares para realizar as estimativas das atividades do projeto. Para cada atividade que o gerente analisar, são exibidos seu prazo e esforço nos projetos similares, bem como o percentual desses valores em relação aos valores totais do projeto. Realizadas as estimativas de todas as atividades do processo de desenvolvimento, as estimativas totais do projeto são calculadas.

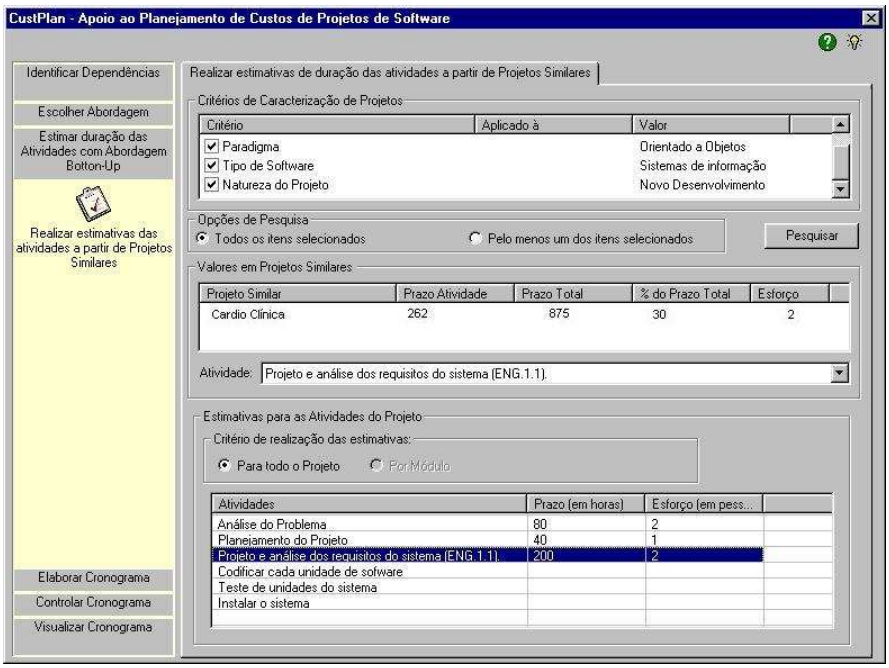


Figura 5.16 – Tela de realização das estimativas do projeto com abordagem *bottom-up*

5.4.1.3 Elaborar Cronograma

Após terem sido determinadas as estimativas de prazo e esforço das atividades do projeto (utilizando-se a abordagem *top-down* ou *bottom-up)*, o cronograma deve ser elaborado. Para isso, o gerente seleciona a atividade *Elaborar o Cronograma* na *CustPlan*.

O primeiro passo para elaborar o cronograma é determinar os caminhos críticos do projeto. Assim, o gerente seleciona a sub-atividade *Identificar Caminhos Críticos.* Quando o gerente acessa essa opção pela primeira vez, os caminhos críticos vêm calculados e identificados em negrito na lista de atividades do projeto. Nas próximas vezes que o gerente acessar a tela, caso tenham ocorrido alterações nas dependências e/ou durações das atividades do processo de desenvolvimento, o botão “Atualizar Caminhos Críticos” estará habilitado para que a lista de caminhos críticos possa ser alterada. Para cada caminho crítico, o gerente pode registrar algumas observações que julgue necessárias.

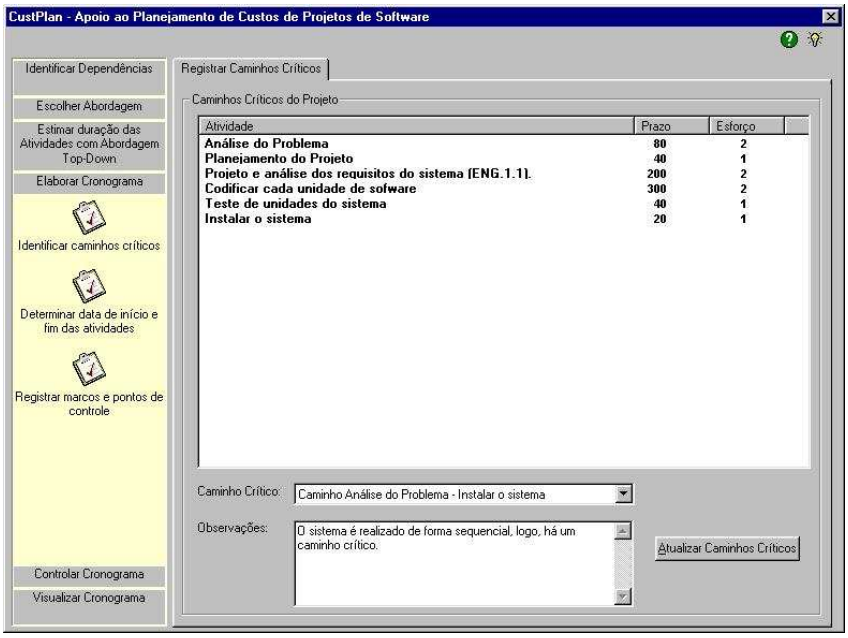


Figura 5.17 – Tela de Identificação dos caminhos críticos

Após identificar os caminhos críticos, as datas de início e fim das atividades devem ser informadas. O gerente seleciona a sub-atividade *Determinar datas de início e fim das atividades* e, analisando os prazos de cada atividade, informa suas datas de início e fim.

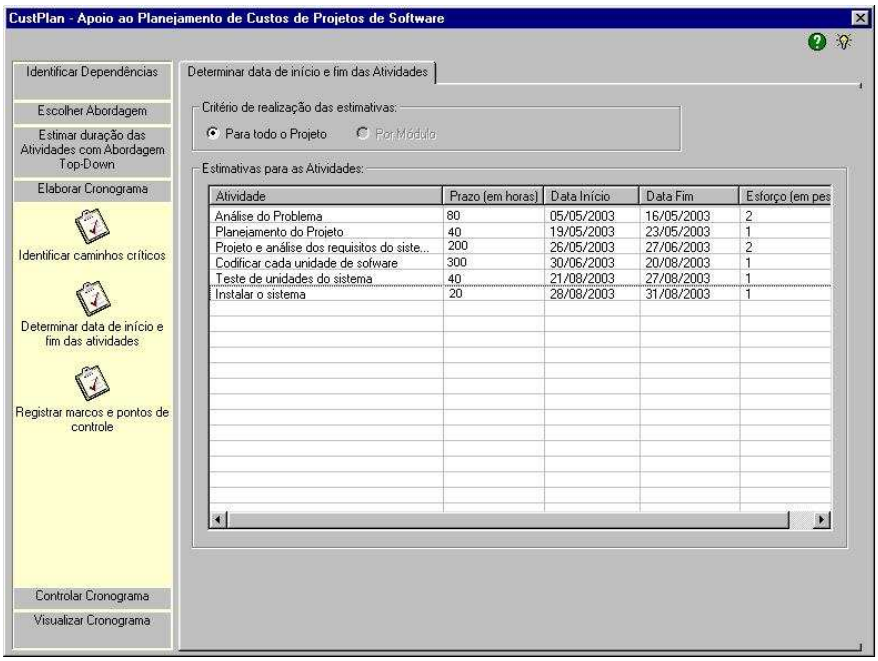


Figura 5.18 – Tela de Determinação das datas das atividades

Para finalizar o cronograma, o gerente seleciona a sub-atividade *Registrar Marcos e Pontos de Controle*. Os marcos e pontos de controle registrados no Plano de Acompanhamento do projeto são, então, registrados no cronograma e indicados por ícones com as letras M e P que indicam se a atividade em questão constitui um marco ou ponto de controle. Quando o gerente acessa essa opção pela primeira vez, os marcos e pontos de controle vêm registrados. Nas próximas vezes que o gerente acessar a tela, caso tenham ocorrido alterações no Plano de Acompanhamento, o botão "Atualizar Marcos e Pontos de Controle" estará habilitado para que as alterações possam ser registradas pelo gerente.

É importante ressaltar que essa funcionalidade ainda está inativa na *CustPlan*, pois o Plano de Acompanhamento está sendo desenvolvido em um projeto de final de curso e ainda não está implementado.

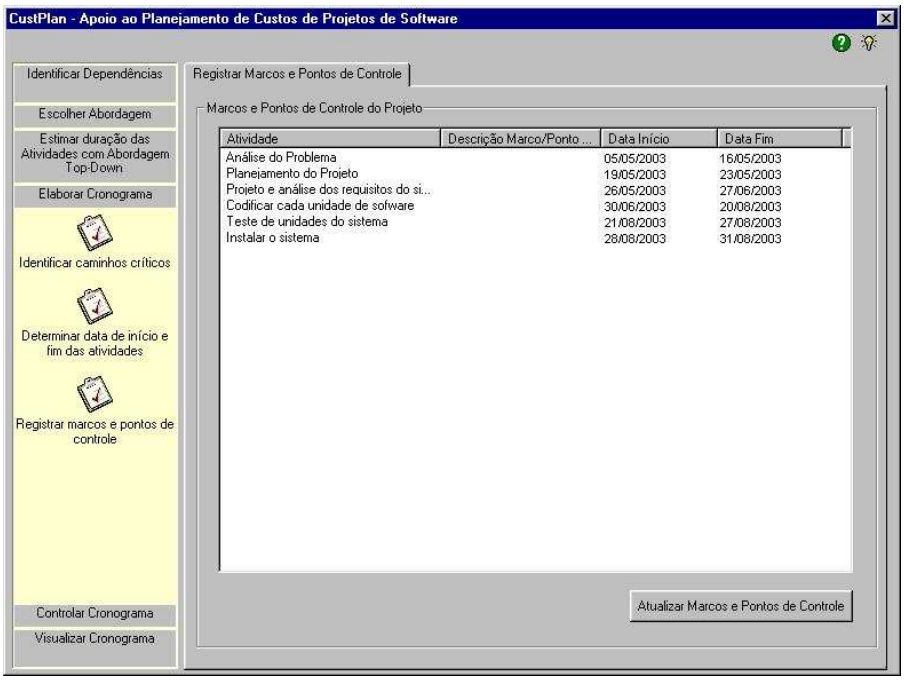


Figura 5.19 – Tela de Registro dos marcos e pontos de controle

Após ser elaborado o cronograma, o gerente pode acessar a atividade *Visualizar Cronograma* e escolher a versão do cronograma que deseja visualizar. É gerado um arquivo html com os dados do cronograma do projeto. A figura 5.20 apresenta a tela utilizada para gerar o cronograma e a figura 5.21 apresenta o arquivo html gerado.

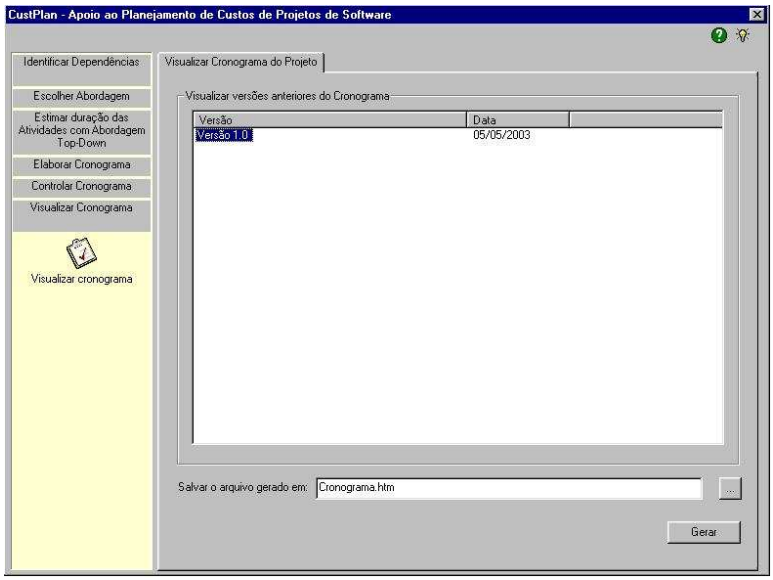
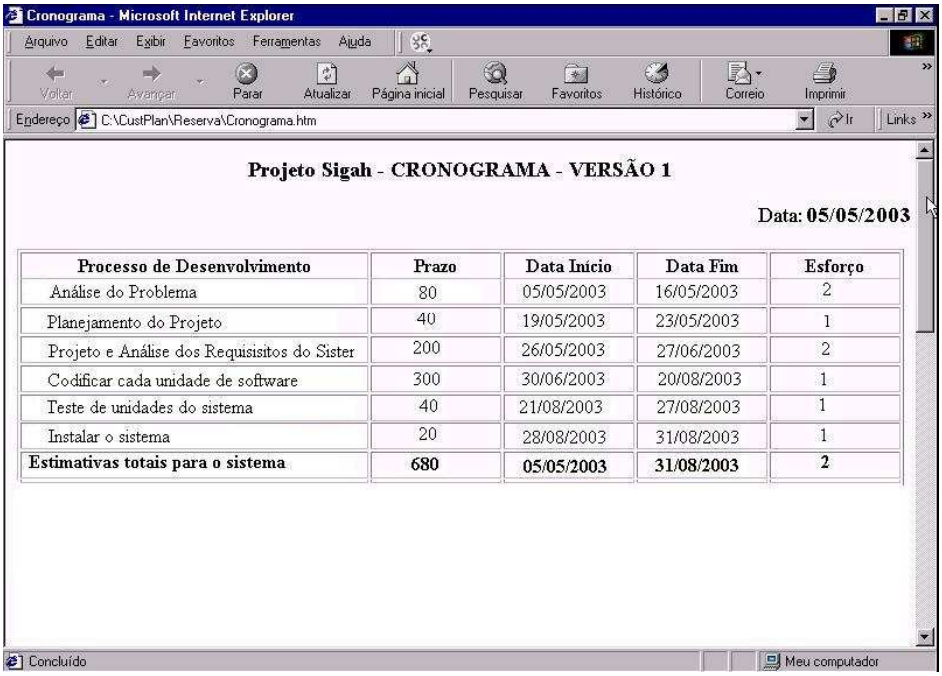


Figura 5.20 – Tela de Visualização do Cronograma



Processo de Desenvolvimento	Prazo	Data Início	Data Fim	Esforço
Análise do Problema	80	05/05/2003	16/05/2003	2
Planejamento do Projeto	40	19/05/2003	23/05/2003	1
Projeto e Análise dos Requisisitos do Sister	200	26/05/2003	27/06/2003	2
Codificar cada unidade de software	300	30/06/2003	20/08/2003	1
Teste de unidades do sistema	40	21/08/2003	27/08/2003	1
Instalar o sistema	20	28/08/2003	31/08/2003	1
Estimativas totais para o sistema	**680**	**05/05/2003**	**31/08/2003**	**2**

Figura 5.21 – Cronograma (arquivo html)

5.4.1.4 Controlar Cronograma

Ao longo do desenvolvimento do projeto deve ser realizado o controle do cronograma. Para isso, na *CustPlan*, o gerente seleciona a atividade *"Controlar Cronograma"* onde serão registrados os desvios que ocorreram no cronograma do projeto e as alterações necessárias.

Para registrar a ocorrência de desvios no cronograma, o gerente seleciona a sub-atividade *Identificar Desvios de Cronograma*. Na tela exibida, o gerente seleciona a atividade do projeto em que o desvio ocorreu e registra a data e o valor em horas do desvio, justificando-o.

No exemplo apresentado na figura 5.22 é possível observar que o gerente registrou um desvio de 8 horas na atividade análise do problema causado por um adiamento na reunião de validação.

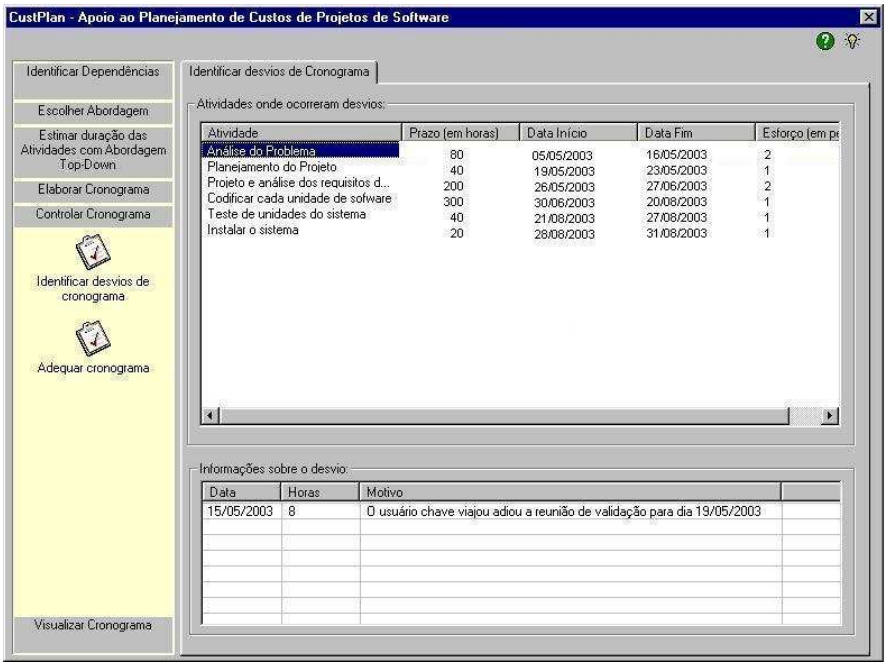


Figura 5.22 – Tela de Identificação dos desvios do cronograma

Identificados os desvios do cronograma, periodicamente, o gerente realiza revisões do cronograma, alterando-o sempre que necessário devido aos desvios registrados. Para realizar a alteração do cronograma, o gerente deve selecionar a sub-atividade *"Adequar Cronograma"*. Na tela que é exibida ao gerente, são apresentadas as atividades do processo

de desenvolvimento do projeto e as atividades que possuem desvios registrados vêm em destaque. Ao selecionar uma atividade do projeto, seus desvios são exibidos e o gerente altera o cronograma analisando os desvios registrados.

No exemplo apresentado no figura 5.23, para alterar o cronograma devido ao desvio de 8 horas registrado para a atividade "Análise do Problema", o gerente decidiu aumentar o prazo dessa atividade em 8 horas e, para não atrasar o projeto, diminuiu o prazo da atividade "Planejamento do Projeto" em 8 horas. É importante observar que as ações corretivas do cronograma são tomadas pelo gerente com base nos dados dos desvios, em sua experiência, no conhecimento organizacional e nas características do projeto.

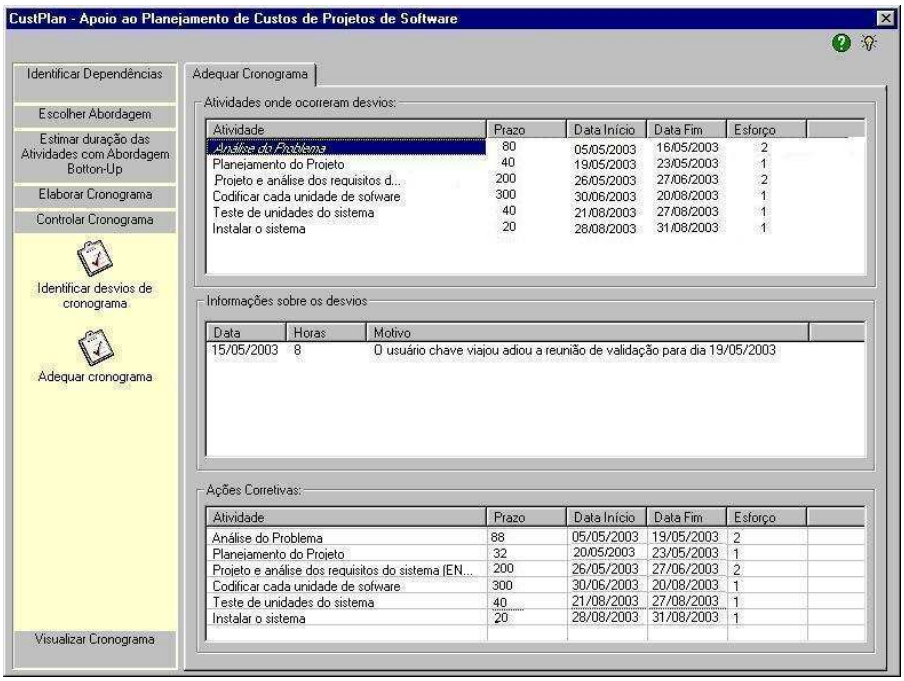


Figura 5.23 – Tela de Alteração do cronograma

Após alterado o cronograma, o gerente pode gerar um novo arquivo através da seleção da atividade *Visualizar Cronograma*, onde ele indica a nova versão a ser visualizada.

O planejamento inicial do tempo é realizado no início do projeto, quando poucas informações são conhecidas, por isso as sub-atividades não são consideradas. Após a especificação de requisitos do sistema estar concluída, deve ser feito o refinamento das

estimativas de tempo, onde as atividades do processo de gerência de tempo aqui apresentadas são executadas novamente, porém, considerando as sub-atividades e as informações obtidas durante a elaboração da especificação de requisitos do sistema.

A figura 5.24 ilustra a estimativa da duração das atividades do projeto no momento do refinamento, ou seja, considerando as sub-atividades do processo de desenvolvimento. Os valores estimados no planejamento inicial podem ser alterados (recalculados) ou apenas expandidos (distribuídos entre as sub-atividades).

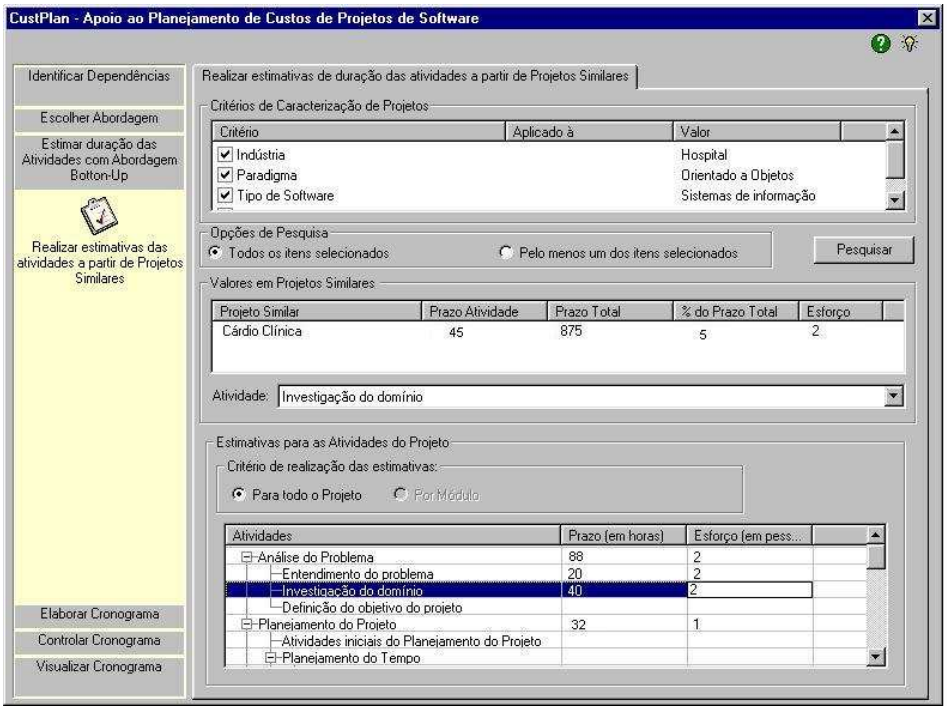


Figura 5.24 – Tela de realização das estimativas das atividades do projeto – Refinamento do Planejamento de Tempo

5.4.2 Planejamento de Custos

Após ser realizado o planejamento do tempo do projeto, utilizando a primeira parte da *CustPlan*, é feita a alocação de recursos humanos ao projeto, que é realizada com o apoio da ferramenta *RHPlan* (SCHNAIDER, 2003).

Após os recursos humanos estarem alocados ao projeto, deve ser elaborado o orçamento do mesmo. Para realizar o planejamento dos custos, *CustPlan* apóia as seguintes atividades: *(i)* Estimar custos; *(ii)* Elaborar o orçamento do projeto; e, *(iii)* Controlar o orçamento do projeto. Durante o planejamento inicial dos custos, apenas as atividades do processo de desenvolvimento são consideradas. As sub-atividades serão consideradas no momento de refinar as estimativas geradas durante o planejamento inicial.

O primeiro acesso à ferramenta para tratar os custos do projeto é feito quando o gerente seleciona, no ADSOrg instanciado, a sub-atividade Planejamento Inicial dos Custos, presente na atividade Planejamento do Projeto. Nesse momento, um ícone da *CustPlan* fica disponível no lado direito da tela.

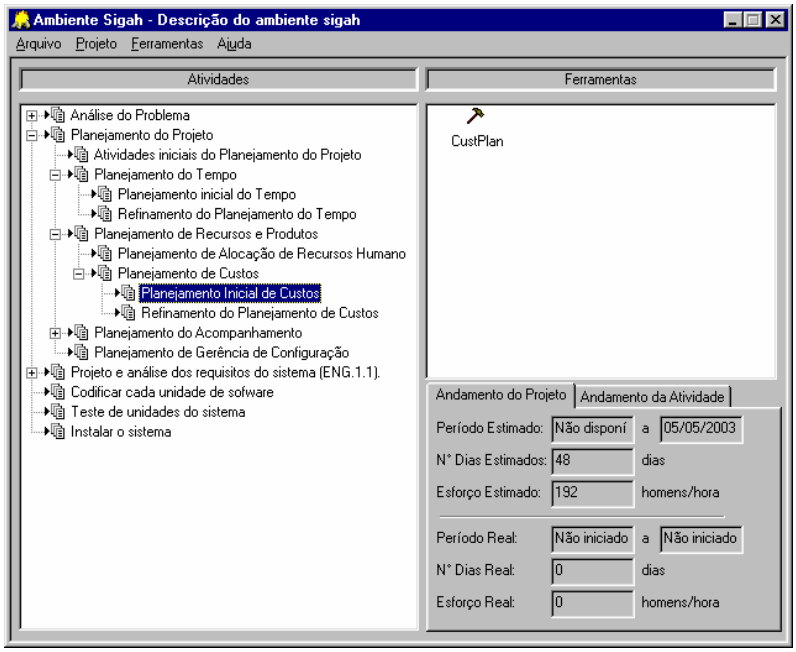


Figura 5.25 - Tela principal de um ADSOrg instanciado com acesso à CustPlan para Planejamento dos Custos

5.4.2.1 Estimar Custos

Para realizar as estimativas dos custos do projeto, o gerente escolhe a atividade *Estimar Custos* na *CustPlan*. O primeiro passo é identificar os elementos de custos. O gerente indica nos *checklists* quais são os recursos que serão utilizados no projeto e sua quantidade. Quando o gerente acessa essa opção o quadro de recursos humanos é preenchido automaticamente, uma vez que foram definidos durante o planejamento de

alocação de recursos humanos. Caso tenham ocorrido alterações no Plano de Alocação de Recursos Humanos, o botão “Atualizar Recursos Registrados em outros Planos” estará habilitado para que o quadro recursos humanos possa ser atualizado considerando as alterações realizadas no Plano de Alocação de Recursos Humanos.

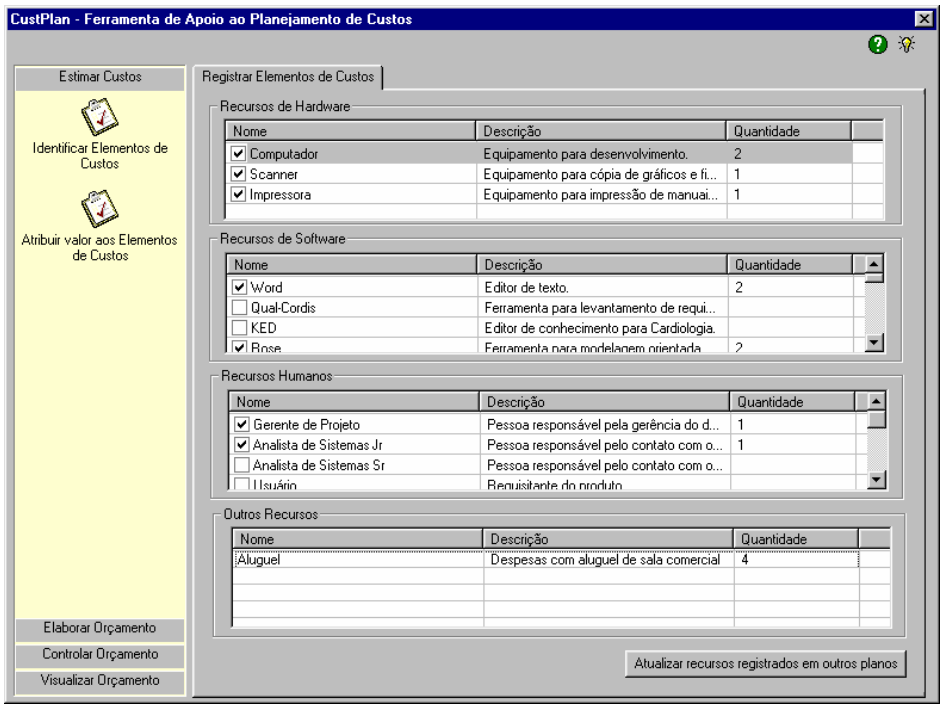


Figura 5.26 - Tela de Identificação dos elementos de custos do projeto

Após terem sido identificados os elementos de custos, o custo destes para o projeto deve ser calculado. Para isso, o gerente seleciona a sub-atividade *Atribuir Valor aos Elementos de Custos* e, para cada elemento de custo, indica o custo unitário. Um custo padrão para a organização é sugerido ao gerente, que pode aceitá-lo ou alterá-lo. No caso de *hardware* e *software*, o custo unitário pode ser o valor de compra ou o valor de uso do recurso (custo agregado ao projeto devido à utilização de recurso já existente, ou seja, que não precisará ser comprado). O gerente pode, ainda, considerar custo zero (0) para alguns elementos, como por exemplo para um *software* que a empresa já possui e que o gerente

não julga necessário incluir o valor de uso como custo para o projeto. Para as despesas (outros recursos), o gerente também deve indicar a frequência (mensal, por exemplo).

O cálculo dos custos com recursos humanos é realizado automaticamente considerando-se seu custo unitário (valor/hora), sua alocação às atividades e o prazo das atividades.

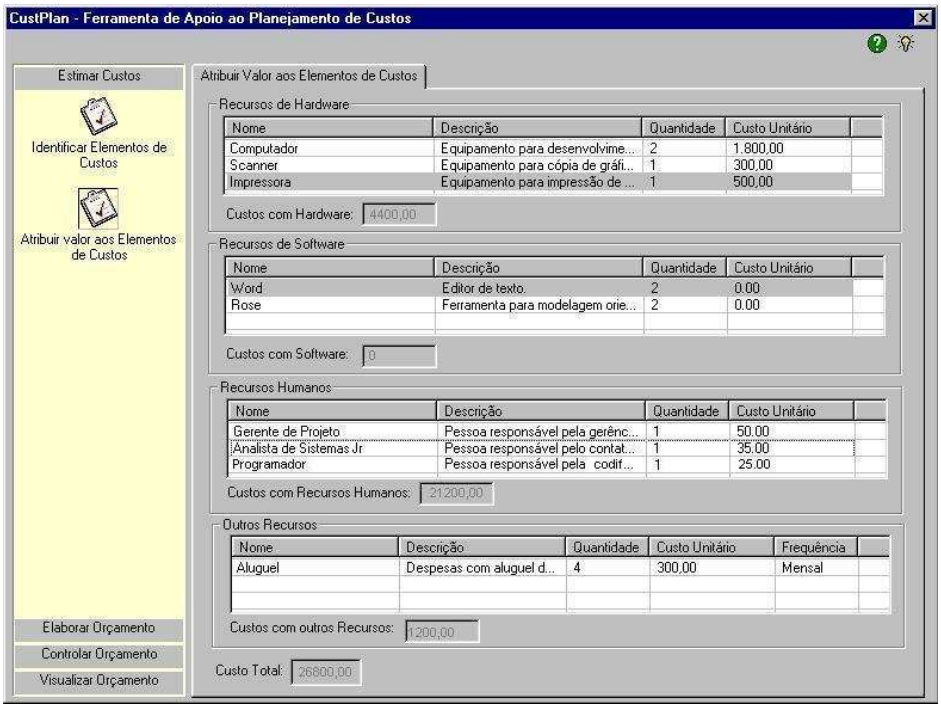


Figura 5.27 - Tela de Atribuição de valor aos elementos de custos do projeto

5.4.2.2 Elaborar Orçamento

Para elaborar o orçamento do projeto, o gerente seleciona a opção *Elaborar Orçamento* na ferramenta *CustPlan*. Em seguida, seleciona a sub-atividade *Registrar Previsão de Receitas*, onde indica a data prevista para cada receita do projeto, seu valor e sua origem.

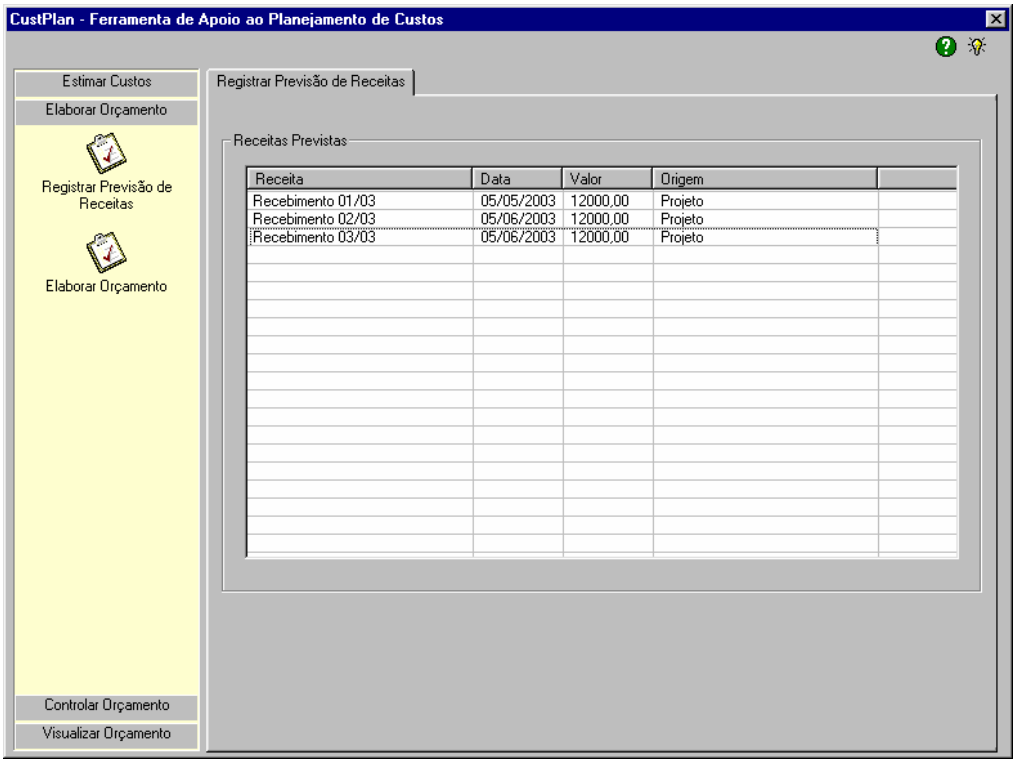


Figura 5.28 - Tela de Registro das Receitas Previstas

Registrada a previsão de receitas, o gerente seleciona a sub-atividade *Elaborar Orçamento*. Os custos com recursos humanos são calculados por atividade do projeto, considerando os dados do cronograma e da alocação de recursos às atividades do projeto. Para os demais recursos, o gerente indica a data em que a despesa ocorrerá. Exemplificando: o gerente identificou o elemento de custo 'aluguel', indicando a quantidade 4. Ao atribuir valor a esse elemento, o gerente registrou um custo de R$ 300,00 com frequência mensal. Para elaborar o orçamento, o gerente deve informar a data em que ocorrerá essa despesa (por exemplo, dia 05/05/2003). Assim, no orçamento será considerado que nos próximos 4 meses, no dia 05, ocorrerá uma despesa de R$ 300,00 referente ao elemento de custo aluguel.

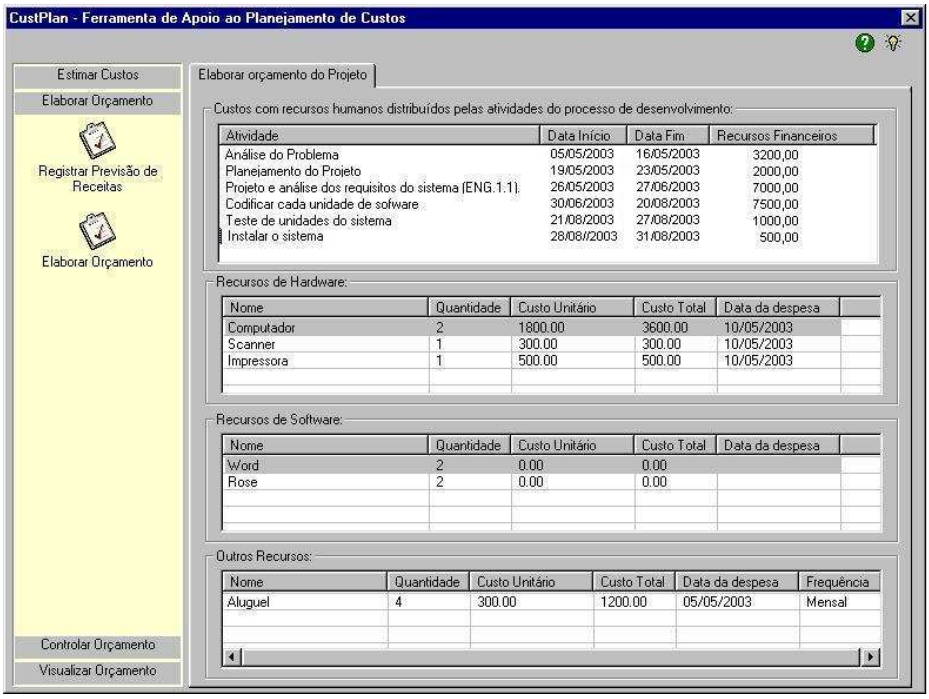


Figura 5.29 - Tela de Elaboração do orçamento

Após ser gerado o orçamento, o gerente pode acessar a atividade *Visualizar Orçamento* para visualizar em arquivo html com os dados do orçamento do projeto.

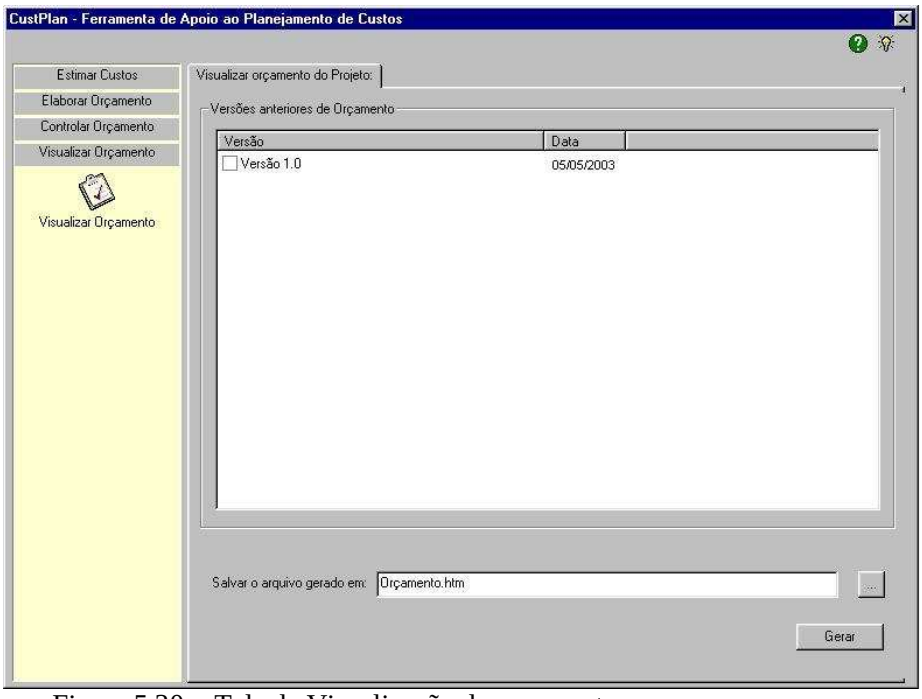
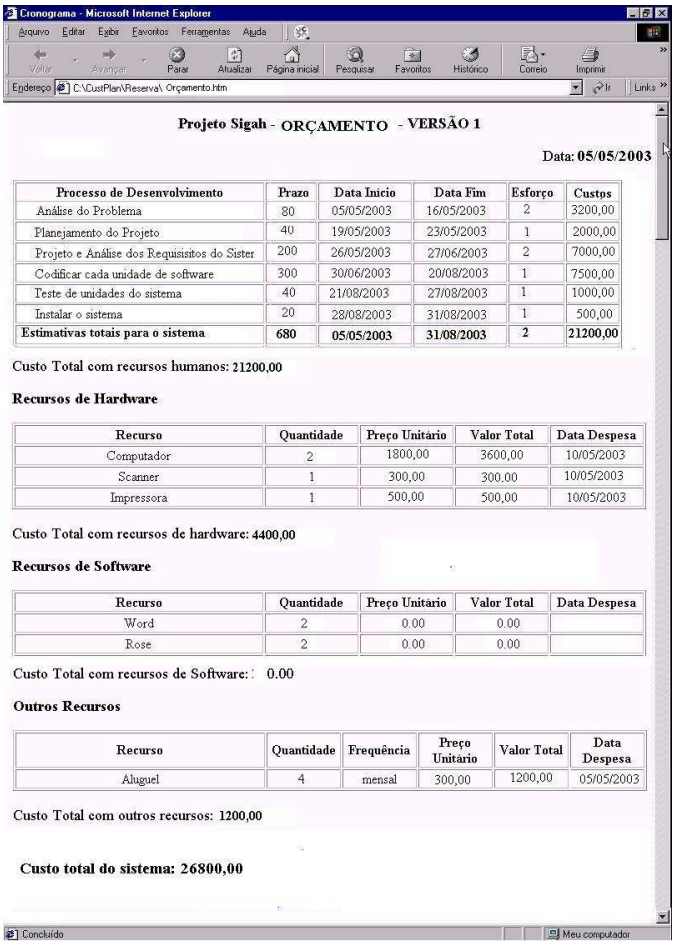


Figura 5.30 - Tela de Visualização do orçamento



Processo de Desenvolvimento	Prazo	Data Início	Data Fim	Esforço	Custos
Análise do Problema	80	05/05/2003	16/05/2003	2	3200,00
Planejamento do Projeto	40	19/05/2003	23/05/2003	1	2000,00
Projeto e Análise dos Requisisitos do Sister	200	26/05/2003	27/06/2003	2	7000,00
Codificar cada unidade de software	300	30/06/2003	20/08/2003	1	7500,00
Teste de unidades do sistema	40	21/08/2003	27/08/2003	1	1000,00
Instalar o sistema	20	28/08/2003	31/08/2003	1	500,00
Estimativas totais para o sistema	**680**	**05/05/2003**	**31/08/2003**	**2**	**21200,00**

Recurso	Quantidade	Preço Unitário	Valor Total	Data Despesa
Computador	2	1800,00	3600,00	10/05/2003
Scanner	1	300,00	300,00	10/05/2003
Impressora	1	500,00	500,00	10/05/2003

Recurso	Quantidade	Preço Unitário	Valor Total	Data Despesa
Word	2	0.00	0.00	
Rose	2	0.00	0.00	

Recurso	Quantidade	Frequência	Preço Unitário	Valor Total	Data Despesa
Aluguel	4	mensal	300,00	1200,00	05/05/2003

Figura 5.31 – Cronograma (arquivo html)

5.4.2.3 Controlar Orçamento

Ao longo do desenvolvimento do projeto, deve ser realizado o controle do orçamento. Para isso, na *CustPlan*, o gerente deve selecionar a atividade *"Controlar Orçamento"*.

O primeiro passo para controlar o orçamento é registrar a realização das receitas previstas. O gerente deve selecionar a sub-atividade *Registrar Receitas Realizadas.* Nesse momento, para cada receita prevista no quadro "Receitas Previstas", o gerente registra no quadro "Receitas Realizadas" a(s) receita(s) realizada(s) correspondentes à prevista, podendo esta ser recebida integralmente ou parcialmente, em diversas sub-receitas. Quando

o valor de uma receita prevista é integralmente recebido, seu status é indicado como *totalmente realizada*.

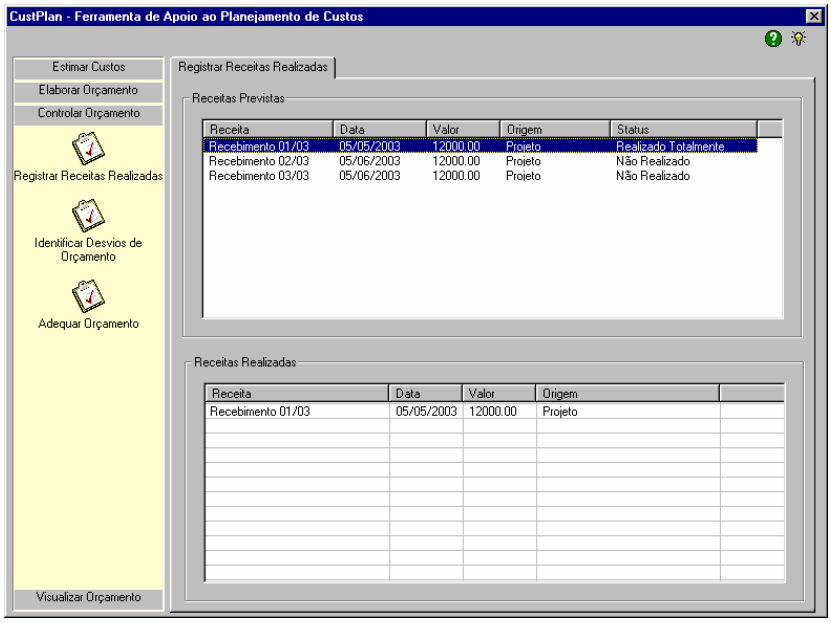


Figura 5.32 - Tela de Registro de receitas realizadas

Caso ocorram desvios no orçamento do projeto, o gerente deve registrá-los. Para registrar a ocorrência de desvios no orçamento, o gerente seleciona a sub-atividade *Identificar Desvios de Orçamento.*

Na tela que é exibida, são disponibilizadas quatro abas, sendo uma para cada tipo de recurso: recursos humanos, recursos de *hardware*, recursos de *software* e outros recursos (despesas em geral). O gerente clica na aba correspondente ao tipo de recurso que gerou a diferença no orçamento para registrá-la. Para os recursos de *hardware*, recursos de *software* e outros recursos, o gerente seleciona o recurso em que o desvio ocorreu e registra a data e o valor do desvio, justificando-o. Na tela da figura 5.31 é exemplificado um desvio referente à compra de uma nova impressora para o projeto.

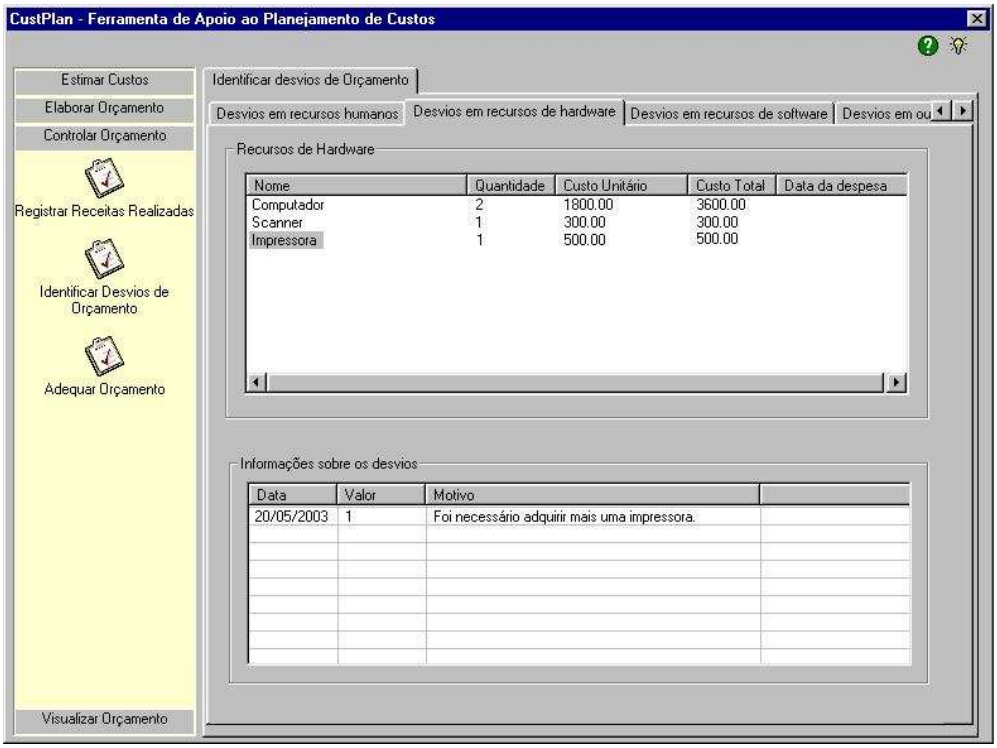


Figura 5.33 – Tela de Identificação de desvios de orçamento provenientes de recursos de *hardware*

É importante observar que sempre que o cronograma do projeto for alterado, o orçamento também deverá ser. Sendo assim, as alterações realizadas no cronograma devem ser refletidas no orçamento. Na tela apresentada na figura 5.34 são exibidas as atividades do projeto e seus respectivos custos. Os desvios de cronograma registrados na parte de tempo da *CustPlan* para as atividades do projeto também estarão registrados para o orçamento. Por exemplo, na figura 5.34 a atividade "Análise do problema" possui um desvio de 8 horas que foi registrado no cronograma (o registro desse desvio foi ilustrado na figura 5.22).

Caso tenha ocorrido algum desvio referente ao custo unitário do recurso humano, este também é registrado neste momento na *CustPlan*.

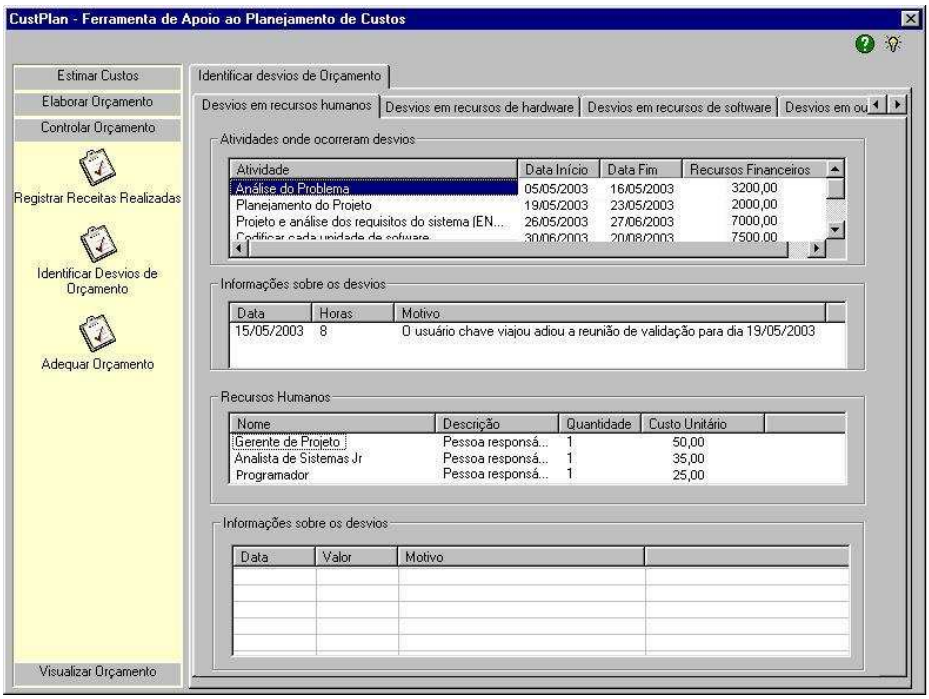


Figura 5.34 – Tela de Identificação de desvios de orçamento provenientes de recursos de humanos

Identificados os desvios do orçamento, o gerente realiza revisões do orçamento, alterando-o sempre que necessário, devido aos desvios registrados. Para realizar a alteração do orçamento, o gerente seleciona a sub-atividade *"Adequar Orçamento"*. Na tela que é exibida ao gerente, são apresentados, em cada aba, os desvios dos recursos e, baseando-se nessas informações, as ações corretivas necessárias são realizadas. No exemplo da tela da figura 5.35, observamos que a ação corretiva para a compra da impressora foi uma alteração na quantidade desse recurso e, consequentemente, em seu custo para o projeto.

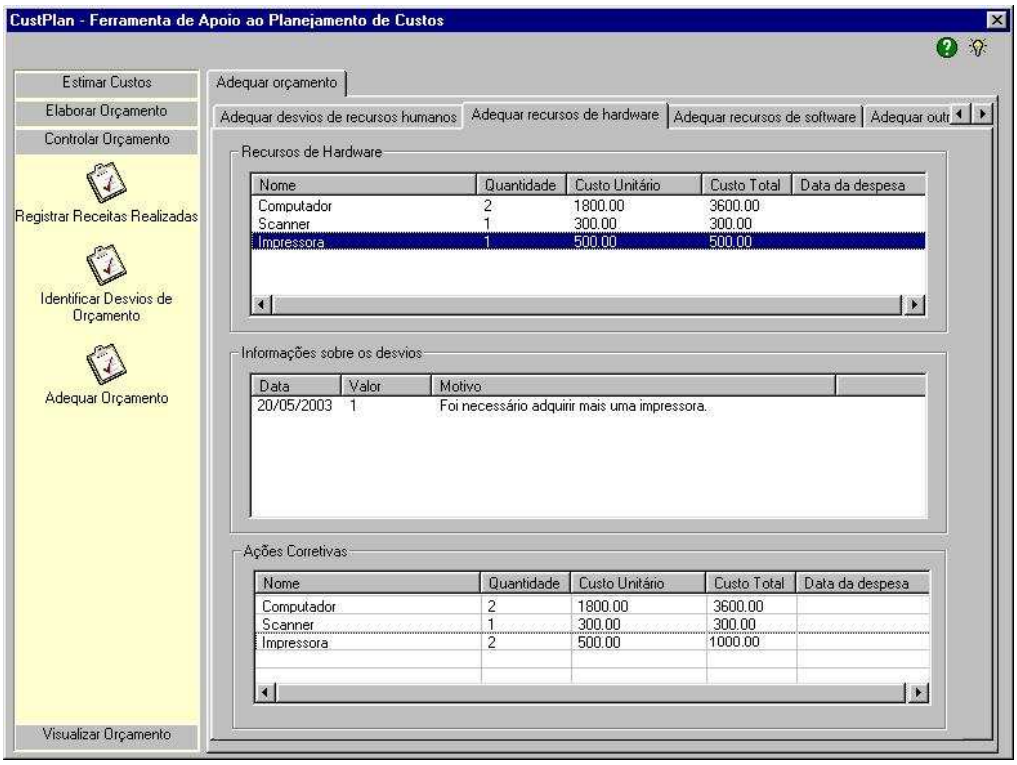


Figura 5.35 – Tela de alteração do orçamento devido a desvios de orçamento provenientes de recursos de *hardware*

As ações corretivas realizadas no cronograma para adequá-lo aos desvios registrados são também registradas no orçamento. Na tela de recursos humanos, no quadro em que são exibidas as atividades, seus custos considerando as ações corretivas são calculados. No exemplo apresentado na tela da figura 5.36, as ações corretivas realizadas pelo gerente para adequar o cronograma ao desvio registrado alteraram os custos das atividades “Análise do Problema” e “Planejamento do Projeto”. Note que o valor acrescido à atividade “Análise do Problema” não equivale ao valor diminuído da atividade “Planejamento do Projeto”, pois os valores dos recursos alocados a essas atividades são diferentes. As alterações de orçamento provenientes das alterações do cronograma são realizadas automaticamente pela *CustPlan*.

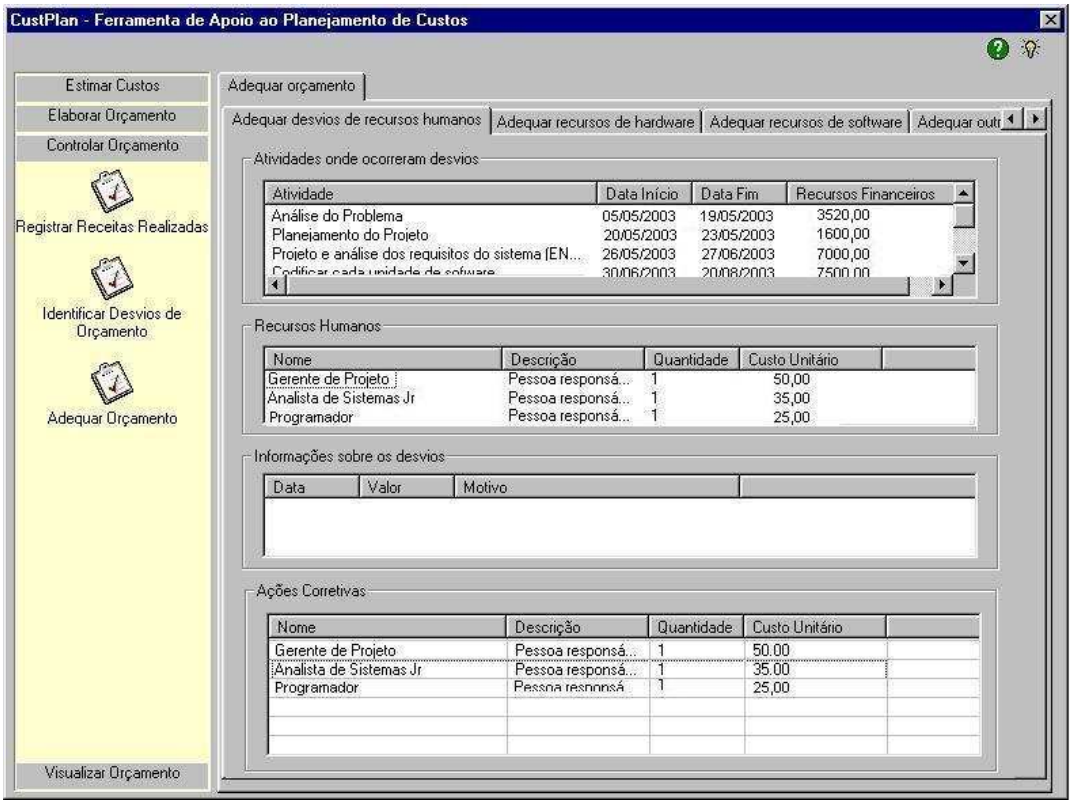


Figura 5.36 - Tela de Alteração do orçamento

Após alterar o orçamento, o gerente pode gerar um novo arquivo através da seleção da atividade *Visualizar Orçamento*.

O planejamento inicial dos custos, assim como o planejamento inicial do tempo, é realizado no início do projeto, quando poucas informações são conhecidas, por isso as sub-atividades não são consideradas. Após a especificação de requisitos do sistema estar concluída, deve ser feito o refinamento das estimativas de tempo e, consequentemente, das estimativas de custos também, onde as atividades aqui apresentadas são executadas novamente, porém, considerando as sub-atividades do processo de desenvolvimento e as informações coletadas durante a elaboração da especificação de requisitos. É importante lembrar que, quando o planejamento de tempo for revisto, o planejamento de custos também deverá ser.

A figura 5.37 ilustra a elaboração do orçamento do projeto no momento do refinamento, ou seja, considerando as sub-atividades do processo de desenvolvimento.

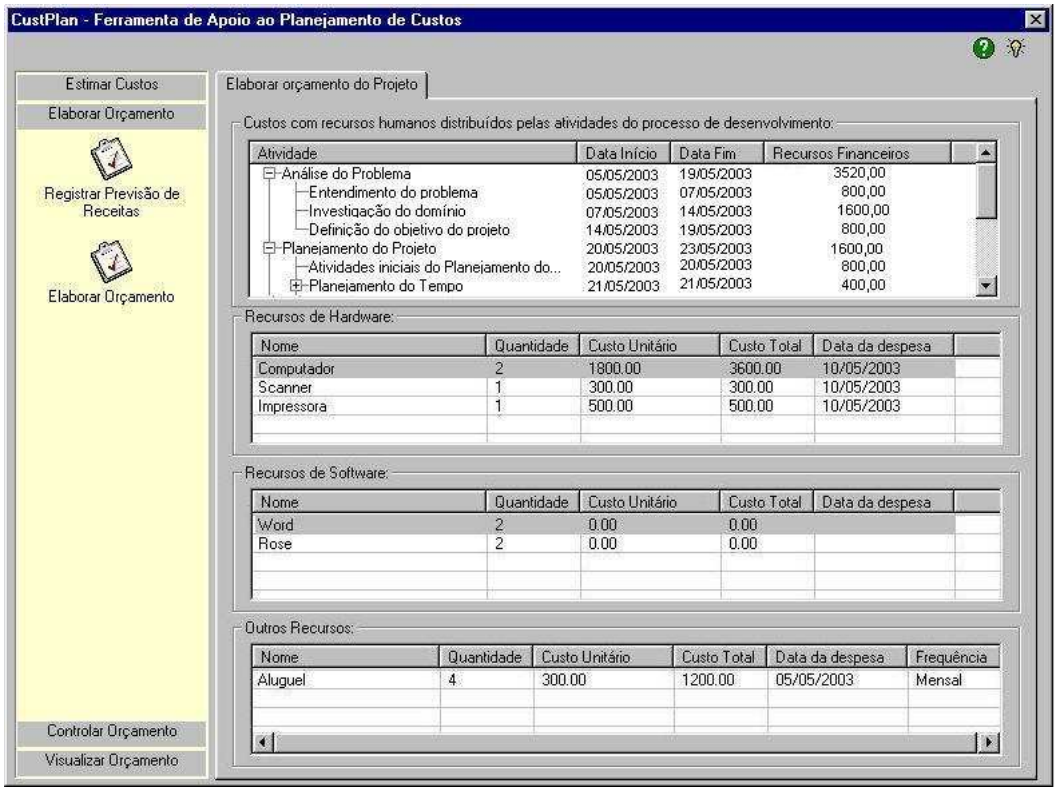


Figura 5.37 - Tela de Elaboração do orçamento – Refinamento das Estimativas de Custos

5.5 Considerações Finais

Neste capítulo foram apresentados os critérios de caracterização da Estação TABA que são utilizados para permitir a recuperação de projetos similares, cujos dados são utilizados na realização de estimativas baseadas em analogia.

A pesquisa realizada para definir um conjunto de dependências usuais entre as atividades do processo de desenvolvimento foi apresentada e a ferramenta *CustPlan,* implementada no contexto do ADSOrg, na Estação TABA, com o objetivo de apoiar o planejamento de tempo e custos, foi descrita e teve sua interface apresentada.

O Anexo 5 apresenta as classes que foram incluídas no modelo da Estação TABA para representar os conceitos envolvidos no planejamento de tempo e custos de projetos.

Capítulo 6

Considerações Finais e Perspectivas Futuras

Neste capítulo são apresentadas as considerações finais desta pesquisa, suas contribuições e suas perspectivas futuras.

6.1 Considerações Finais

A tecnologia hoje disponível para o desenvolvimento de *software* permite, e até induz, a utilização de arquiteturas de sistemas cada vez maiores e mais complexas. Em contrapartida, o prazo para o desenvolvimento de tais produtos tem sido compactado, refletindo a evolução tecnológica e necessidades econômicas do mercado. As dimensões dos produtos de *software* estão atingindo níveis quantitativos cada vez maiores, não ocorrendo o mesmo com a tolerância a falhas de previsão de custos e cronogramas. Esse cenário exige que processos formalizados de gerência de *software* sejam definidos e utilizados (TRINDADE, *et al.* 1999).

A gerência dos prazos e custos de projetos de *software* é muito importante, uma vez que pesquisas mostraram que a maioria dos projetos que fracassam têm como seu principal motivo o mal planejamento dos custos e cronograma. Para auxiliar os gerentes de projeto a gerenciarem os prazos e custos dos projetos, o PMBOK (2000) dedica um capítulo à gerência de tempo e um capítulo à gerência de custos, fornecendo diretrizes e sugestões de ferramentas que auxiliam a gerência dos prazos e custos do projeto e comprovando a importância mundial dessas atividades na realização de projetos. A norma NBR ISO 10006 (2000) também trata os processos de gerência de tempo e gerência de custos, comprovando mais uma vez a importância dessas atividades no contexto da gerência de projetos.

Dada a importância da utilização de métodos eficientes para realizar o planejamento de prazos e custos, muitos pesquisadores dedicaram, e ainda dedicam, pesquisas a essa área, propondo-se a realizar estudos de caso com as técnicas mais conhecidas e, analisando os resultados obtidos nesses estudos, refiná-las e/ou definir novas técnicas e caminhos para realizar as estimativas com sucesso.

MIRANDA (2001) afirmou que técnicas e abordagens eficientes existem. O que não é muito comum, ainda, é a utilização correta dessas técnicas. Esse fato acarreta na elaboração de orçamentos e cronogramas falhos e irreais, que podem contribuir para o fracasso de um projeto.

Prover uma organização de capacidade para utilizar uma abordagem de planejamento de prazos e custos prática e eficaz é um ponto diferencial para a mesma diante das exigências mercadológicas atuais. O trabalho aqui descrito veio, exatamente, propor uma abordagem para o planejamento de prazos e custos nas organizações. Essa abordagem está inserida em um tipo especial de ambientes de desenvolvimento de *software*, chamados de Ambientes de Desenvolvimento de *Software* Orientados à Organização (ADSOrg). Este trabalho encontra-se, assim, no contexto do ADSOrg, um projeto de longo prazo da COPPE, englobando outras ferramentas de gerência de projetos com enfoque em gerência do conhecimento e memória organizacional

Dentre as principais contribuições deste trabalho destacam-se:

(1) A definição de processos de gerência de tempo e gerência de custos baseados na gerência do conhecimento organizacional e nas recomendações da norma NBR ISO 10006 (2000), do relatório técnico 16326 da ISO/IEC 12207(1998) e no PMBOK (2000);

(2) A proposta de uma caracterização para os projetos da Estação TABA. Assim, os projetos podem ser agrupados como similares de acordo com critérios genéricos e/ou específicos.

(3) A utilização da notação para diagramas de workflow para fornecer uma representação para os processos definidos, destacando-se as atividades, suas entradas, saídas e elementos interagentes. A utilização da notação contribuiu para observar seus pontos fortes e fracos.

(4) A disponibilização de uma ferramenta para auxiliar a elaboração do Plano de Custos e Cronograma em Ambientes de Desenvolvimento de *Software* Orientados à Organização, utilizando métodos paramétricos de estimativas e os princípios da gerência do conhecimento e Memória Organizacional.

(5) A definição de um conjunto de dependências usuais entre as atividades do processo de desenvolvimento de *software*.

Os benefícios da abordagem proposta poderão ser avaliados em um procedimento de validação da ferramenta. Porém, a validação de uma ferramenta como a *CustPlan* implica em sua utilização em vários projetos o que excede em muito o tempo esperado para uma tese de mestrado. Portanto, a validação será realizada no contexto do projeto ADSOrg englobando outras ferramentas de gerência de projetos.

A abordagem proposta introduz o planejamento de custos na Estação TABA e representa um passo importante na construção de ferramentas de apoio à gerência de projetos centradas na utilização do conhecimento organizacional. Tal perspectiva traz grandes benefícios para as organizações de *software* e é uma das principais tendências na área de Engenharia de *Software*.

6.2 Perspectivas Futuras

Buscando-se melhorar e expandir a abordagem de planejamento de custos proposta, algumas perspectivas de trabalhos futuros são destacadas.

Inicialmente, a definição e implementação da técnica de busca por projetos similares. Atualmente, os projetos estão sendo recuperados baseando-se em uma busca exata, ou seja, dois projetos são considerados similares somente quando todos os seus critérios de caracterização são idênticos. A definição de uma técnica de busca por projetos similares é fundamental para permitir a utilização eficiente do conhecimento acumulado por uma organização. Algumas técnicas estudadas foram citadas nos capítulos dois e cinco, porém, seria interessante um estudo detalhado das abordagens encontradas na literatura.

Uma outra melhoria seria a disponibilização dos dados armazenados no repositório da organização referentes a projetos similares de forma mais expressiva para o gerente do projeto, para auxiliá-lo nas estimativas de novos projetos. Um exemplo desses dados seria os valores médios de prazo, esforço e custos utilizados em projetos classificados como pequenos, médios e grandes.

Outra melhoria seria a implementação do processo de calibração do modelo COCOMO II, para que o mesmo possa se adequar, cada vez mais, à organização em que está sendo utilizado. Para realizar a calibração do COCOMO II é necessário que o repositório da organização contenha dados de diversos projetos. Sendo assim, a realização

da calibração depende da validação e efetiva utilização do ADSOrg para que esses dados possam existir.

Uma outra perspectiva bastante interessante seria a definição de uma ontologia de prazos e custos, inserida na ontologia de Engenharia de *Software*, que é um dos componentes da estrutura de conhecimento de ADSOrg. A ontologia de prazos e custos tornaria possível a definição de um vocabulário comum no que diz respeito ao conhecimento de custos da organização.

Outra melhoria muito importante é a implementação da integração da *CustPlan* com duas ferramentas existentes no TABA: com *RiscPlan* (FARIAS, 2002) onde a ocorrência de um risco exige a execução de um plano de contingência que provocará desvios no cronograma planejado; e com a planilha de acompanhamento de projeto, onde os recursos humanos do projeto registram o andamento das atividades que estão desenvolvendo. Assim, desvios de atraso no desenvolvimento dessas atividades poderiam ser registrados automaticamente. É importante observar que essa melhoria seria realizada para apoiar a atividade de controle do cronograma do projeto no processo de gerência de tempo.

Para apoiar a disseminação e utilização do conhecimento, poderia ser desenvolvida uma consulta pró-ativa ao conhecimento, onde a própria *CustPlan* seria capaz de sugerir ao gerente dados de projetos anteriores sem, necessariamente, ter sido executada uma busca por projetos similares para isso.

A abordagem proposta neste trabalho define processos para a gerência de tempo e custos e uma ferramenta que apóia a realização desses processos na Estação TABA. A proposta de processos e ferramentas que apóiem o desenvolvimento de *software* reforça a percepção de que este não mais pode ser visto como uma prática artesanal ou individual, pois suas especificidades o caracterizam como uma área complexa e abrangente e, como tal, deve ser planejado e controlado para que se obtenha um *feedback*, no mínimo, satisfatório.

Referências Bibliográficas

ABECKER, A., BERNADI, A., HINKELMANN, K. *et al.*, 2001, *"Toward a Technology for Organizational Memories"*, IEEE Intelligent Systems, v. 13, May/June, pp. 40-48.

ABRAN, A., JACQUET, J.-P., 1999, *"A Structured Analysis of the New ISO Standard on Functional Size Measurement-definition of Concepts"* Software Engineering Standards. Fourth IEEE International Symposium and Forum, pp. 230 –241.

ACKERMAN, M., HALVERSON, C., 2000, *"Reexamining Organizational Memory"*, Communications of the ACM, v. 43, n. 1, Jan, pp. 59-64.

ALAVI, M., LEIDNER, D., 1999, *" Knowledge Management Systems: Emerging Views and Practices from the Field"*, In: 32th Hawaii International Conference on System Sciences, IEEE Intelligent Systems, pp. 1-11.

ANGELIS, L., STAMELOS, I., MORISIO, M., 2001, *"Building a Software Cost Estimation Model Based on Categorical Data"*, Software Metrics Symposium - METRICS, pp. 4 –15.

ARAÚJO, M. A. P., 1998, *"Automatização do Processo de Desenvolvimento de Software nos Ambientes Instanciados pela Estação TABA"*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

ARMOUR, P., 2002, *"Then Unmyths of Project Estimation"*, Communications of the ACM, vol 45, nº 11, Nov, pp. 15-18.

BASILI, V., CALDIERA, G., ROMBACH, D., 1994, *"The Experience Factory"*. In: John Marcianiak, Volume 1 of Encyclopedia of *Software* Engineering, Chapter X, John Wiley & Sons.

BASILI, V., NETO, M. G. M., SEAMAN, C. B., KIM, Y. M., 2001a *"A Prototype Experience Management System for a Software Consulting Organization"*, 13th International Conference on *Software* Engineering and Knowledge Engineering – SEKE, pp. 29-36.

BASILI, V, LINDVALL,M., COSTA, P. , 2001b, *"Implementing the Experience Factory concepts as a set of Experience Bases"*, 13th International Conference on *Software* Engineering and Knowledge Engineering – SEKE , pp. 102-109.

BECERRA-FERNANDEZ, I., 1998, *"Center for Innovation and Knowledge Management"*, SIGGROUP Bulletin, v. 19, Apr, pp. 46-51.

BIELAK, J., 2000, *"Improving Size Estimates Using Historical Data"*, IEEE Software , Volume: 17 Issue: 6 , Nov/Dec, pp. 27 –35.

BIGGAN, J., 2001, *"Defining Knowledge: An Epistemological Foundation for Knowledge Management"*, In: 34th Hawaii International Conference on System Sciences, IEEE Intelligent Systems, pp. 1-7.

BIRK, A., DINGSOYR, T., STALHANE, T., 2002, *"Postmortem: Never Leave a Project Without It"*, IEEE Software , Volume: 19 Issue: 3 , May/Jun, pp. 43 –45.

BOEHM, B. W., 1981, *"Software Engineering Economics"*, Prentice Hall, Upper Saddle River.

BOEHM, B. W., ABTS, C., BROWN, A.W., CHULANI, S., CLARK, B.K., HOROWITZ, E., MADACHY, R., REIFER, D., STEECE, B., 2000, *"Software Cost Estimation with COCOMO II"*, Prentice Hall.

BOEHM, B. W., 2000, **"***Safe and Simple Software Cost Analisys"*, IEEE Software, Sep/Oct, pp. 14-17.

BOEHM, B.W., FAIRLEY, R. E., 2000, *" Software Estimation Perspectives"*, IEEE Software , Volume: 17 Issue: 6 , Nov/Dec, pp. 22 –26.

BRAGA, A. , 1996, *"Análise de Pontos de Função"*, IBPI Press, Rio de Janeiro, RJ.

BRIAND, L.C., EMAM, K., SURMANN, D., WIECZOREK, I., MAXWELL, K.D., 1999, *"An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques"*, Communications of the ACM, May, pp. 313-322.

BRIAND, L.C., LANGLEY, T., WIECZORECK, I., 2000, *"A replicated Assessment and Comparison of Common Software Cost Modeling Techniques"*, Communications of the ACM, Jun, pp. 377-386.

BROOKS, P. F., 1975, *"The Mythical Man-Month"*, Addison Wesley, pp. 14-26, 88-94.

CHANDRASEKARAN, B., et al., 1993, *"Task-Structure Analysis for Knowledge Modeling"*. In: B. V. Cuena (ed.), Knowledge Oriented Software Design, Elsevier Science Publishers**.** Ref: VILLELA (et al. 2000).

CHANDRASEKARAN, B., JOSEPHSON, J. R., BENJAMINS V. R., 1999, *"What Are Ontologies, and Why Do We Need Them?"*, IEEE Intelligent Systems & their applications, v. 14, n. 1, Jan/Feb, pp. 20-26.

CHRISTENSEN, M. J., THAYER H. R., 2001, *"The Project Managers: Guide to Software Engineering's Best Pratices"*, Ed. IEEE Computer Society.

CRUZ, C. D., 1998, *"Em Direção a um Modelo de Custos de Desenvolvimento de Software Orientado a Objetos"*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

DAVENPORT, T., DE LONG, D., BEERS, M., 1998. *"Successful Knowledge Management Projects*", Sloan Management Review, v. 39, n. 2, pp. 43-57.

DEKKERS, C.A., 1999, *"Determination of Functional Domains for Use with Functional Size Measurement - Opportunities to Classify Software From a Business Perspective"*, Software Engineering Standards. Fourth IEEE International Symposium and Forum, pp. 227 –229.

DONALDSON, S. E., SIEGEL, S. G., 2000, *"Successful Software Development"*, Prentice Hall, chapter 2, pp. 63-118.

EMAM, K., DROUIN, J., MELO W., 1998, SPICE – *The Theory and Practice of Software Process Improvement and Capability Determination*, IEEE Computer Society Press.

ENGELKAMP,S., HARTKOPT, S., BROSSLER, P., 2000, *"Project Experience Database: A Report Based on First Practical Experience"*, PROFES - Product Focused *Software* Process Improvement, pp. 204-215.

FARIAS, L. L., ROCHA, A., TRAVASSOS, G., 2001, *"Planejamento de Riscos em Ambientes de Desenvolvimento de Software Orientados à Organização"*, WorkShop de Teses - XV Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro-RJ.

FARIAS, L. L., 2002, *"Planejamento de Riscos em Ambientes de Desenvolvimento de Software Orientados à Organização"*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FENTON, N. E.,1991, *"Software Metrics: A Rigorous Approach"*, Chapman and Hall.

FISCHER, G, OSTWALD, J., 2001, *"Knowledge Management: Problems, Promises, Realities and Challenges"*, IEEE Intelligent Systems, Jan/Feb, pp. 60-72.

FORD, B., 1976, *" Missing Data Procedures: A Comparative Study"*, Proc. Social Statistics Section, pp. 324-329.

GARMUS, D., HERRON, D., 2001, *"Function Point Analysis: Measurement Practices for Successful Software Projects"*, Addison Wesley.

GOMES, A.G.J., 2001, *"Avaliação de Processos de Software Baseada em Medições"*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

GRAY, A.R., MACDONELL, S.G., SHEPPERD, M.J., 1999, *"Factors Systematically Associated With Errors in Subjective Estimates of Software Development Effort: The Stability of Expert Judgment"*, Software Metrics Symposium - METRICS, pp. 216 –227.

HASTINGS, T. E., SAJEEV, A. S. M., 2001, *"A Vector-Based Approach to Software Size Measurement and Effort Estimation"*, IEEE Transactions on *Software* Engineering v. 27, n. 4, Apr, pp. 337-350.

HENDRIKS, P., VRIENS, D., 1999, *"Knowledge-based systems and knowledge management: Friends or foes?"*, Information & Management, v. 35, n. 2, Feb, pp. 113-125.

HOST, M., WOHLIN, C., 1997, *"A Subjective Effort Estimation Experiment", I*nformation and Software Technology, v. 39, pp. 755-762.

HUMPREY, W., 1995, *"A Discipline for Software Engineering"*, Addison Wesley.

IDRI, A., ABRAN, A., 2001, *"A Fuzzy Logic Based Set of Measures for Software Projects Similarity: Validation and Possible Improvements"*, Software Metrics Symposium - METRICS, pp. 85-96.

ISO/IEC DTR 16326 – *Software Engineering – Guide for the Application of ISO /IEC 12207 to Project Management*, 1999.

ISO/IEC 14143 – *Information Technology – Software Measurement – Functional Size Measurement, 1998.*

IVES, B., GIFFORD, T., HANKINS, D., 1998, *"Integrating Learning Through Knowledge (&Skills) Management"*, SIGGROUP Bulletin, v. 19, n. 1, Apr, pp. 51-55.

JEFFERY, R..,RUHE, M., WIEEZOREK, I., 2001, *"Using Public Domain Metrics to Estimate Software Development Effort"* , 7th International *Software* Metrics Simposium 2001 – Metrics 2001, pp. 16-27.

JOHNSON, P. M., MOORE, C. A., DANE, J. A., BREWER, R. S., 2000, *"Empirically Guided Software Effort Guesstimation"*, IEEE Software, Nov/Dec, pp. 51-56.

JONES, M., 1998, *"Knowledge Management at Work: Current Approaches and Case Studies"*, SIGGROUP Bulletin, v. 19, n. 1, Apr, pp. 33-34.

JONES, C., 1999, *"Software Sizing"*, IEEE Review, v.45, Issue: 4 , Jul, pp. 165-167

JONES, C., 2000, *"Software Assessments, Benchmarks, and Best Practices"*, Addison-Wesley Information Technology Series, pp. 657 .

JORGENSEN, M., INDAHL, U., SJOBERG, D., 2001, *"Software Effort Estimation by Analogy and "Regression Toward the Mean"*, 13th International Conference on *Software* Engeeering and Knowledge Engineering – SEKE , pp. 268-274.

JOSHI, K. D., 2001, *"A Framework to Study Knowledge Behaviors During Decision Making"*, In: 34 th Hawaii International Conference on System Science, IEEE, pp. 1-12.

KROMREY, J., HINES, C., 1994, *"Nonrandomly Missing Data in Multiple Regression: An Empirical Comparation of Commom Missing Data Trearments"*, Education and Psycological Measurement, vol. 54, nº 3, pp. 573-593.

KURTZ, T., 2001, *"Ask Pete, Software Planning and Estimation Through Project Characterization"*, Requirements Engineering. Fifth IEEE International Symposium, pp. 286 –287.

KUSUMOTO, S., INOUE, K., KASIMOTO, T., SUZUKI, A., YUURA, K., TSUDA, M., 2000, "*Function Point Measurement for Object-oriented Requirements Specification"*, Computer Software and Applications Conference - COMPSAC, pp. 543 –548.

LEE, J., KIM, Y., YU, S., 2001, "*Stage Model for Knowledge Management*", In: 34th Hawaii International Conference on System Sciences, IEEE Software, pp. 1-10.

LIEBOWITZ, J., 2002, *"A Look at NASA Goddard Space Flight Center's Knowledge Management Iniciatives"*, IEEE Software , v. 19, Issue: 3 , May/Jun, pp. 40 –42.

LITTLE, R., RUBIN, D., 1987, *'Statistical Analisys with Missing Data"*, John Wilwy and Sons.

MARKKULA, M., 1999, "*Knowledge Management in Software Engineering Projects*", In: Proceedings of the 11th International Conference on Software Engineering & Knowledge Engineering, Kaiserslautern, Germany, Jun, pp. 20-27.

MACHADO, L., ROCHA, A., 1999, "*Modelo para Definição, Especialização e Instanciação de Processos de Software*". In: Anais do Workshop de Teses em Engenharia de Software - XIII Simpósio Brasileiro de Engenharia de Software, Florianópolis, Brasil, Out, pp. 43-47.

MACHADO, L., 2000, *"Modelo para Definição, Especialização e Instanciação de Processos de Software na Estação TABA"*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MAIDANTCHIK, C., 1999, *"Modelo de Processo de Gerência de Software para Equipes Geograficamente Dispersas"*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MENDES, E., COUNSELL, S., 2000, *"Web Development Effort Estimation Using Analogy"*, Software Engineering Conference, pp. 203 –212.

MENDONÇA, M. G., SEAMAN, C.B., BASILI, V., KIM, Y., 2001, *"A Prototype Experience Management System for a Software Consulting Organization"*, Software Engineering and Knowledge Engineering – SEKE, Buenos Aires, Argentina, June.

MENZIES, T., SINSEL, E., 2000, *"Practical Large Scale what-if Queries: Case Studies with Software Risk Assessment"*, Automated Software Engineering. Fifteenth IEEE International Conference, pp. 165 –173.

MIRANDA, E., 2001, *"Improving Subjective Estimates Using Paired Comparisons"*, IEEE Software , Volume: 18 Issue: 1 , Jan/Feb, pp. 87 –91.

MONTONI, M., ROCHA, A. R., TRAVASSOS, G. H., 2002, "*Aquisição de Conhecimento nos Processos de Negócio*", In: 7o Workshop de Teses em Engenharia de Software, Gramado, Brasil, Outubro.

MOSES, J., CLIFFORD, J., 2000, *"Learning How to Improve Effort Estimation in Small Software Development Companies"*, Computer Software and Applications Conference - COMPSAC, pp. 522 –527.

MURCH, R., 2000, *"Project Management: Best Pratices for IT Professionals"*, Prentice Hall.

NAGESWARAN, S., 2001, "*Test Effort Estimation Using Use Case Points*", Quality Week, San Francisco, California, EUA, Jun, pp.1-6.

NBR ISO 10006 – *Gestão da Qualidade: Diretrizes para Qualidade no Gerenciamento de Projetos,* 2000.

NBR ISO/IEC 12207 – *Tecnologia de Informação – Ciclos de Vida de Software*, 1998.

OHLSSON, M.C., WOHLIN, C., 1999, *"An Empirical Study of Effort Estimation During Project Execution"*, Software Metrics Symposium - METRICS, pp. 91 –98.

O'LEARY, D. E., 1998a, *"Enterprise Knowledge Management"*, IEEE Intelligent Systems, Mar, pp. 54-61.

O'LEARY, D. E., 1998b, *"Using AI in Knowledge Management: Knowledge Bases and Ontologies"*, IEEE Intelligent Systems, v. 13, n. 3, May/Jun, pp. 34-39.

O'LEARY, D. E., 1998c, *"Using AI in Knowledge Management: Knowledge Bases and Ontologies"*, IEEE Intelligent Systems, v. 13, n. 3, May/Jun, pp. 34-39.

O'LEARY, D.E., STUDER, R., 2001, *"Knowledge Management: An Interdisciplinary Approach"*, IEEE Intelligent Systems, Jan/Feb, pp. 24-25.

O'LEARY, D.E., 2001, *"How Knowledge Reuse Informs Effective System Design and Implementation"*, IEEE Intelligent Systems, Jan/Feb, pp. 44-49.

OLIVEIRA, K., 1999, *"Modelo para Constrição de Ambientes de Desenvolvimento de Software Orientados a Domínio"*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

PAULK, M. C., WEBER, C. V., CURTIS, B., CHRISSIS, M. B. (eds.), 1995, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Carnegie Mellon University, *Software* Engineering Institute, Addison-Wesley Longman Inc.

PRADO, D., 1998, *"PERT/CPM"*, Série Gerência de Projetos, volume 4, 2ª edição, Ed. de Desenvolvimento Gerencial,.

PRIETO-DIAZ, R., 1991, *"Implementing faceted classification for software reuse"*, Communications of the ACM, vol.34, no.5, May, pp. 89 – 97.

PMBOK – Project Management Body of Knowledge, 2000, PMI – Project Management Institute.

PREECE, A., FLET, A., SLEEMAN, A., CURRY, D., MEANY, N., PERRY, P., 2001, *"Better Knowledge Management through Knowledge Engineering"*, IEEE Intelligent Systems, Jan/Feb, pp. 36-43.

PRESSMAN, R. S., 2001, *"Software Engineering: A Practitioner's Approach"*, McGraw-Hill International Editions, 5th ed.

PUTNAM, L.H.,1978, *"A General Empirical Solution to the Macro Software Sizing and Estimation Problem"*, IEEE Systems, Jul.

RAINER, A., SHEPPERD, M., 1999, "*Re-planning for a Successful Project Schedule*", Software Metrics Symposium - METRICS, pp. 72 –81.

RAMASUBRAMANIAN, S., JAGADEESAN, G., 2002, *"Knowledge Management at Infosys"*, IEEE Software, May/Jun, pp. 53-55.

RAMESH, B., 2002, *"Process Knowledge Management with Traceability"*, IEEE Software, v. 19, Issue: 3 , May/Jun, pp. 50 –52.

RAYMOND, M., 1987, *"A Comparisonof Methods for Treating Incomplete Data in Selection Research"*, Education and Psycological Measurement, vol. 47, pp. 13-26.

REIFER, J. D., 2002, *A Little Bit of Knowledge is a Dangerous Thing"*, IEEE Software, May/Jun, pp. 14-15.

ROCHA, A.R.C, AGUIAR T. C., SOUZA, J. M., 1990, *"TABA: A Heuristic Workstation for Software Development"*, In: Proceedings of COMPEURO'90, Tel Aviv, Israel, Maio.

ROCHA, A. R. C., 2000, Gerência de Projetos, Notas de Aula.

RUNESON, P., BORGQUIST, N., LANDIN, M., BOLANOWSKI, W., *2000, "An Evaluation of Functional Size Methods and a Bespoke Estimation Method for Real-Time Systems"*, PROFES – Product Focused *Software* Process Improvement, pp 339-352.

RUS, I., LINDVALL, M., 2002, *"Knowledge Management in Software Engineering"* , IEEE Software , v. 19, Issue: 3 , May/Jun, pp. 26 –38.

SCHNAIDER, L., 2003, *"Planejamento de Alocação de Recursos Humanos em Ambientes de Desenvolvimento de Software Orientados à Organização"*, Tese de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

SEPPÃNEN, V., KOMI-SIRVIÕ, S., MÃNTYNIEMI, A., 2002, *"Toward a Practical Solution for Capturing Knowledge for Software Projects"*, IEEE Software, May/Jun, pp. 60-62.

SHEPPERD, M., SCHOFIELD, C., 1997, *"Estimating Software Project Effort Using Analogies"*, Software Engineering, IEEE Transactions on , v. 23, Issue: 11 , Nov 1997, pp. 736 –743.

SHNEIDER, K., HUNNIUS, J., BASILI, V. R., 2002, *"Experience in Implementing a Learning Software Organization"*, IEEE Software, May/Jun, pp. 46-49.

SKYRME, D., 1998, *"Knowledge Management Solutions - The IT Contribution"*, SIGGROUP Bulletin, v. 19, n. 1, Apr, pp. 34-39.

STAAB, S., STUDER, R., SCHNURR, H.P., SURE, Y., 2001, *"Knowledge Management and Ontologies"*, IEEE Intelligent Systems, Jan/Feb, pp. 26-34.

STRIKE, K., EMAM, K. E., MADHAVJI, N., 2001, *"Software Cost Estimation with Incomplete Data"*, IEEE Transactions on Software Engineering, vol. 27, nº 10, Oct, pp. 890-908.

TRAVASSOS, G. H., 1994, *"O Modelo de Integração de Ferramentas da Estação TABA"*, Tese de D. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

TRINDADE, A. L. P., PESSOA,M. S. P., SPINOLA, M. M. , 1999, *"COCOMO II – Uma Compilação de Informações sobre a Nova Métrica"*, V Congresso Internacional de Engenharia Informática da Universidade de Buenos Aires – Argentina, pp. 1-17.

UEMURA, T., KUSUMOTO, S., INOUE, K., 1999, *"Function Point Measurement Tool for UML Design Specification"*, Software Metrics Symposium - METRICS, pp. 62 –69.

VILLELA, K., TRAVASSOS, G.H., ROCHA, A.R., 2000, *"Ambientes de Desenvolvimento de Software Orientados à Organização"*, Publicação Técnica COPPE/UFRJ - ES530/00 Rio de Janeiro, RJ, Abril.

VILLELA, K., TRAVASSOS, G.H., ROCHA, A.R., 2001a *"Ambientes de Desenvolvimento de Software Orientados a Organização"*, IDEAS'2001 - Workshop Ibero-americano de Ingeniería de Requisitos y Ambientes de *Software*, Jan Jose, Costa Rica, Abril.

VILLELA, K., SANTOS, G., BONFIM, C., et al., 2001b, *"Knowledge Management in Software Development Environments"*, 14th International Conference Software & Systems Engineering and their Applications", Paris, Dezembro.

VILLELA, K., SANTOS, G., TRAVASSOS, G. H., ROCHA, A. R., 2002, *"Gestão de Conhecimento em Ambientes de Desenvolvimento de Software"*, 2ª Jornada Ibero-Americana de Engenharia de Software e Engenharia de Conhecimento, Salvador, Brasil, Outubro.

WANGENHEIM, C. G. V., LICHTNOW, D., WANGENHEIM , A.. V., 2001, "*A Hybrid Approach for Corporate Memory Management Systems in Software R&D Organizations"*, 13th International Conference on *Software* Engineering and Knowledge Engineering – SEKE 2001 , pp. 326-330.

WEBER, C. K., ROCHA, A. R. C., NASCIMENTO, C. J., 2001, "*Qualidade e Produtividade em Software",* 4ª Edição, Ed. Makron Books.

WEI, C., HU, P. J., CHEN, H., 2002, "*Design and Evaluation of a Knowledge Management System"*, IEEE Software, May/Jun, pp. 56-59.

WIIG, K., 1994, "*Knowledge Management - The Central Focus for Intelligent acting Organizations*", Schema Press. Ref: SKYRME (1998).

YAMAURA, T., KIKUNO, T., 1999, "*A Framework for Top-down Cost Estimation of Software Development",* Computer Software and Applications Conference – COMPSAC, pp. 322 –323.

Anexo 1

Realização de Estimativas utilizando Análise de Pontos de Função e Pontos de Caso de Uso

Este anexo apresenta os passos necessários para realizar estimativas utilizando as técnicas Análise de Pontos de Função, Pontos de Caso de Uso e um exemplo de utilização da técnica Análise de Pontos de Função.

A1.1 O Processo de Contagem de Pontos de Função

O processo de contagem dos pontos de função pode ser dividido em sete etapas: *(i)* determinar tipo de contagem; *(ii)* identificar a fronteira da aplicação; *(iii)* contar as funções tipo dados; *(iv)* contar as funções tipo transação; *(v*) calcular pontos de função não ajustados (com base nos resultados obtidos em *(iii)* e *(iv)*); *(vi)* calcular o valor do fator de ajuste; e *(vii)* calcular os pontos de função ajustados (com base nos resultados obtidos em *(v)* e *(vi)*) , como mostra a figura A1.1. A execução dessas etapas é descrita a seguir.

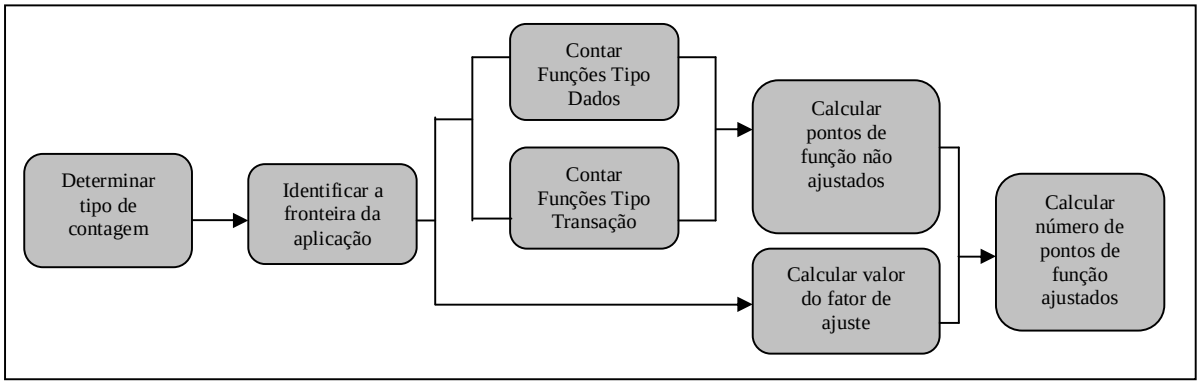


Figura A1.1 – Visão Geral do Processo de Análise de Pontos de Função (GARMUS e HERRON, 2001)

(i) *Determinar o Tipo de Contagem*

Para realizar a contagem dos pontos de função de um projeto, inicialmente, é preciso determinar o tipo de contagem a ser realizada, podendo esta ser:

- *Projeto de Desenvolvimento:* mede a funcionalidade fornecida aos usuários finais do *software* para a primeira instalação da aplicação. Inclui as

funcionalidades da contagem inicial da aplicação e as funcionalidades requeridas para conversão de dados.

- *Projeto de Manutenção:* mede as modificações realizadas para aplicações existentes. Inclui as funcionalidades fornecidas aos usuários através de adição, modificação ou exclusão de funções na aplicação. As funcionalidades de conversão de dados também devem ser consideradas, caso existam. Após a manutenção, a contagem da aplicação deve ser refeita para refletir as alterações realizadas.
- *Aplicação:* mede uma aplicação instalada. É também referenciada como contagem de linha de base ou contagem instalada e avalia as funcionalidades correntes providas aos usuários finais da aplicação.

(ii) I*dentificar a Fronteira da Aplicação*

Após determinado o tipo de contagem, a fronteira da aplicação deve ser identificada. Ela indica a separação entre o projeto que está sendo medido e as aplicações externas ao domínio do usuário. É através dela que torna-se possível definir quais funcionalidades serão incluídas no processo de contagem dos pontos de função.

(iii) *Contar Funções Tipo Dados*

Nesta etapa as funcionalidades da aplicação começam a ser identificadas e contadas.

A funcionalidade da aplicação é avaliada em termos do *quê* é fornecido pela mesma, não do *como* é fornecido. Apenas componentes definidos e solicitados pelo usuário devem ser contados (GARMUS e HERRON, 2001).

As Funções Tipo Dados representam as funcionalidades fornecidas pelo sistema ao usuário, para atender às necessidades referentes aos dados que o sistema irá manipular. Essas funções podem ser:

1) *Arquivo Lógico Interno (ALI) :* grupo logicamente relacionado de dados ou informações de controle, identificável pelo usuário, mantido dentro da fronteira da aplicação que está sendo controlada. Por exemplo: as tabelas ou classes do sistema.

2) Arquivo de Interface Externa (AIE): grupo logicamente relacionado de dados ou informações de controle, referenciado pela aplicação, identificável pelo usuário, mantido fora da fronteira da aplicação que está sendo controlada. Por exemplo: as tabelas acessadas em um outro sistema.

A diferença básica entre um ALI e um AIE é que o último **não** é mantido pela aplicação que está sendo contada. Um AIE contado para uma aplicação sempre será contado como um ALI em sua aplicação de origem.

Nas definições de ALI e AIE foram utilizados alguns termos e expressões que merecem esclarecimento. São elas:

- *Informações de Controle:* são dados utilizados pela aplicação para garantir aderência com os requisitos funcionais especificados pelo usuário. Por exemplo: datas e horas são utilizadas pelos usuários para estabelecer a sequência ou o momento de eventos. Assim, datas e horas são informações de controle.
- *Identificável pelo Usuário:* refere-se aos requisitos específicos que um usuário ou grupo de usuários seria capaz de definir para a aplicação.
- *Mantido:* refere-se ao fato de que o dado pode ser modificado através de um processo elementar da aplicação. Um processo elementar é a menor atividade capaz de produzir resultados significativos para o usuário. Por exemplo: incluir, alterar e excluir.

Cada Arquivo Lógico Interno e cada Arquivo de Interface Externa possui dois tipos de elementos que devem ser contados para cada função identificada:

- *Tipos de Elementos de Dados (TED):* campo único, reconhecido pelo usuário, não recursivo. Por exemplo: campos das tabelas.
- *Tipos de Elementos de Registros (TER):* subgrupo de dados, reconhecido pelo usuário. Por exemplo: generalização/especialização de classes.

Ao final dessa etapa devem estar identificados quantos Arquivos Lógicos Internos e Arquivos de Interface Externa o sistema possui e para eles, quantos são os Tipos de Elementos de Dados e os Tipos de Registros encontrados.

(iv) *Contar Funções Tipo Transação*

As Funções Tipo Transação representam as funcionalidades de processamento dos dados fornecidas pelo sistema ao usuário. Essas funções podem ser:

1) *Entrada Externa (EE):* processo elementar da aplicação que processa dados ou informações de controle que vêm de fora da fronteira da aplicação que está sendo controlada. Exemplos: validações, fórmulas e cálculos matemáticos cujos parâmetros vêm de fora da fronteira da aplicação.
2) *Saída Externa (SE):* processo elementar da aplicação que gera dados ou informações de controle que são enviados para fora da fronteira da aplicação que está sendo controlada. Exemplos: relatórios e gráficos.
3) *Consulta Externa (CE):* processo elementar da aplicação que representa uma combinação de entrada (solicitação de informação) e saída (recuperação de informação). Exemplos: consultas implícitas, verificação de senhas e recuperação de dados com base em parâmetros.

Cada Entrada Externa, Saída Externa e Consulta Externa possui dois tipos de elementos que devem ser contados para cada função identificada:

- *Tipos de Elementos de Dados (TED):* campo único, reconhecido pelo usuário, não recursivo. Por exemplo: campos das tabelas.
- *Tipos de Arquivos Referenciados ou Arquivos Referenciados (TAR):* arquivos lógicos utilizados para processar a entrada e/ou saída. É o total de ALI e AIE utilizados pela transação.

Ao final dessa etapa devem estar identificadas quantas Entradas Externas, Saídas Externas e Consultas Externas o sistema possui e, para elas, quantos são os Tipos de Elementos de Dados e os Arquivos Referenciados encontrados.

(v) *Calcular os Pontos de Função Não Ajustados*

Após serem contadas todas as Funções Tipo Dados e as Funções Tipo Transação e seus elementos, é preciso calcular os pontos de função não ajustados, que refletem especificamente as funcionalidades fornecidas ao usuário pelo produto. Para isso, é preciso identificar a complexidade e a contribuição, em pontos por função, de cada uma das funções e elementos contados.

Para determinar a complexidade e contribuição das funções e seus elementos, é necessário utilizar as relações dos valores de complexidade e contribuição fornecidas pela técnica. A seguir são apresentadas tabelas que indicam a complexidade e contribuição das funções e seus elementos em um sistema, de acordo com a contagem estabelecida nas etapas *(iii)* e *(iv)*.

A tabela A1.1 indica a complexidade de um Arquivo Lógico Interno ou Arquivo de Interface Externa de acordo com o número de Tipos de Elementos de Dados e de Tipos de Elementos de Registros identificados para ele.

		Tipos de Elementos de Dados		
Tipos de Elementos de Registros		***1 a 19***	***20 a 50***	***≥51***
	1	*BAIXA*	*BAIXA*	*MÉDIA*
	2 a 5	*BAIXA*	*MÉDIA*	*ALTA*
	≥6	*MÉDIA*	*ALTA*	*ALTA*

Tabela A1.1 – Complexidade de Arquivos Lógicos Internos e Arquivos de Interface Externa
(GARMUS e HERRON, 2001)

A tabela A1.2 indica a complexidade de uma Entrada Externa de acordo com o número de Tipos de Elementos de Dados e de Arquivos Referenciados identificados para ela. Também é utilizada para determinar a complexidade das entradas de uma Consulta Externa.

		Tipos de Elementos de Dados		
Tipos de Arquivos Referenciados		***1 a 4***	***5 a 15***	***≥16***
	0 a1	*BAIXA*	*BAIXA*	*MÉDIA*
	2	*BAIXA*	*MÉDIA*	*ALTA*
	≥3	*MÉDIA*	*ALTA*	*ALTA*

Tabela A1.2 – Complexidade de Entradas Externas e Entradas das Consultas Externas
(GARMUS e HERRON, 2001)

A tabela A1.3 indica a complexidade de uma Saída Externa de acordo com o número de Tipos de Elementos de Dados e de Arquivos Referenciados identificados para ela. Também é utilizada para determinar a complexidade das saídas de uma Consulta Externa.

		Tipos de Elementos de Dados		
Tipos de Arquivos Referenciados		***1 a 5***	***6 a 19***	***≥20***
	0 a1	*BAIXA*	*BAIXA*	*MÉDIA*
	2 a 3	*BAIXA*	*MÉDIA*	*ALTA*
	≥4	*MÉDIA*	*ALTA*	*ALTA*

Tabela A1.3 - Complexidade de Saídas Externas e Saídas das Consultas Externas

(GARMUS e HERRON, 2001)

A tabela A1.4 indica as contribuições (pesos) obtidas através das complexidades calculadas para as funções identificadas.

	Contribuições (pesos)				
Complexidades	***ALI***	***AIE***	***EE***	***SE***	***CE***
BAIXA	7	5	3	4	3
MÉDIA	10	7	4	5	4
ALTA	15	10	6	7	6

Tabela A1.4 - Contribuições (pesos) das complexidades

(GARMUS e HERRON, 2001)

Para calcular os pontos de função não ajustados, multiplica-se o número de funções identificadas para uma determinada complexidade por sua contribuição. Ao final, soma-se todos os pontos de função encontrados.

A seguir é apresentado um exemplo para o cálculo dos pontos de função não ajustados (PFNA) gerados pelos ALI de um sistema hipotético. O mesmo deve ser feito para a outras funções do sistema (AIE, EE, SE e CE).

Função	***Itens Contados por Complexidade***	***Contribuição***	***Total por Complexidade***	***Total de PFNA da Função***
ALI	1 Baixa	x 7	7	42
	2 Média	x 10	20	
	1 Alta	x 15	15	

Tabela A1.5 – Exemplo de cálculo dos pontos de função não ajustados

(vi) *Calcular Valor do Fator de Ajuste*

O número de pontos de função não ajustados de um sistema reflete a funcionalidade que o sistema fornecerá ao usuário, sem considerar as especificidades do sistema. Por exemplo, um mesmo sistema pode ser implementado para operar *stand alone* para um cliente e em arquitetura cliente servidor para outro. As funcionalidades seriam as mesmas, o que resultaria na mesma contagem de pontos de função não ajustados, mas quando considera-se as características do sistema para cada cliente, observa-se que os pontos de função devem ser ajustados para refletir a maior complexidade do sistema na arquitetura cliente servidor.

Para ajustar os pontos de função encontrados na etapa *(v)* devem ser levadas em consideração 14 (quatorze) características do sistema que serão analisadas e fornecerão o valor do fator de ajuste. São elas: Comunicação de Dados, Processamento Distribuído, Performance, Configuração Altamente Utilizada, Taxa de Transações, Entrada de Dados On-Line, Eficiência do Usuário Final, Atualização *On-Line*, Processamento Complexo, Reutilização, Facilidade de Operação, Facilidade de Instalação, Múltiplos Locais e Modificações Facilitadas.

Para cada característica deve ser atribuído um nível de influência de 0 (zero) a 5 (cinco), onde 0 (zero) indica nenhuma influência, 1 (um) influência mínima, 2 (dois) influência moderada, 3 (três) influência média, 4 (quatro) influência significativa e 5 (cinco) grande influência.

Para calcular o valor do fator de ajuste deve-se seguir a relação

$$VFA = (GIT * 0{,}01) + 0{,}65$$

Onde

- VFA é o valor do fator de ajuste
- GIT é o grau de influência total (soma de todos os valores dos níveis de influência).

(vii) *Calcular Pontos de Função Ajustados*

Após calculado o valor do fator de ajuste, os pontos de função não ajustados serão ajustados, multiplicando-se o valor dos pontos de função não ajustados (PFNA), obtidos em *(v),* pelo valor do fator de ajuste (VFA),obtido em *(vi)*.

Assim,

$$PFA = PFNA \times VFA$$

O número de pontos de função encontrado representa o tamanho da aplicação de acordo com sua funcionalidade.

Para calcular as estimativas de esforço, prazo e custos para a aplicação é necessário conhecer valores como o custo de um ponto de função (por exemplo R$90,00)[7] e o tempo necessário para realizar um ponto de função (por exemplo 2,5 h), ou o esforço para realizar um ponto de função (por exemplo 14 pessoas/mês) e o custo do esforço. Com esses valores é possível calcular as estimativas para o projeto através das relações entre o número total de pontos de função do sistema e os valores de um ponto de função.

Para determinar os valores de um ponto de função, a organização pode realizar medições em projetos anteriores e obter um valor médio para o ponto de função. Caso não existam projetos anteriores podem ser consultadas tabelas disponibilizadas pelo IFPUG *(Institute Function Point Users Group)* e por seus órgãos representantes em cada país.

A1. 2 O processo de contagem de Pontos de Caso de Uso

O processo de contagem dos pontos de caso de uso pode ser dividido em cinco etapas: *(i)* calcular o peso dos atores da aplicação; *(ii)* calcular o peso dos casos de uso da aplicação; *(iii)* calcular pontos de caso de uso não ajustados (com base nos resultados obtidos em *(i)* e *(ii)*); *(iv)* calcular o valor dos fatores de ajuste; e, *(v)* calcular os pontos de caso de uso ajustados (com base nos resultados obtidos em *(iii)* e *(vii)*). A execução dessas etapas é descrita a seguir.

(i) *Calcular o Peso dos Atores da Aplicação*

[7] Valor fornecido pela empresa FATTO Consultoria e Sistemas – Vitória - ES

A primeira etapa da contagem de pontos de caso de uso é classificar os atores envolvidos em cada caso de uso do sistema e calcular seu peso. Um ator pode ser classificado como *simples, médio* ou *complexo,* de acordo com a tabela A1.6.

Tipo de ator	***Peso***	***Descrição***
Simples	1	Outro sistema acessado através de uma API de programação.
Médio	2	Outro sistema interagindo através de um protocolo de comunicação.
Complexo	3	Usuário interagindo através de uma interface gráfica.

Tabela A1.6 – *Pesos dos Atores (NAGESWARAN, 2001)*

O peso dos atores (PA) equivale à soma dos pesos de todos os atores do sistema. Desta forma, uma aplicação projetada para dois tipos de usuários (gerente e usuário comum) e que seja acessada por um outro sistema, utilizando um protocolo de comunicação, por exemplo, terá o valor de PA igual a oito (dois atores de nível *complexo* e um ator de nível *médio*.

(ii) *Calcular o Peso dos Casos de Uso da Aplicação*

Após ser calculado o peso dos atores, é preciso calcular o peso dos casos de uso do sistema. Os casos de uso, assim como os atores, são classificados em três tipos: *simples, médio* ou *complexo.* A classificação de um caso de uso se dá de acordo com o número de transações envolvidas em seu processamento. Por transação , entende-se como uma série de processos que devem, garantidamente, ser realizados em conjunto, ou cancelados em sua totalidade, caso não seja possível completar o processamento. O peso dos casos de uso (PCU) equivale à soma dos pesos de todos os casos de uso do sistema. A tabela A1.7 apresenta a classificação dos casos de uso segundo o número de transações.

Tipo de Caso de Uso	***Número de Transações***	***Peso***
Simples	Até 3	1
Médio	4 a 7	2
Complexo	Mais de 7	3

Tabela A1.7 – *Pesos dos Casos de Uso de acordo com o número de transações (NAGESWARAN, 2001)*

Os casos de uso podem também ser classificados de acordo com o número de classes envolvidas, como mostra a tabela A1.8.

Tipo de Caso de Uso	***Número de Transações***	***Peso***
Simples	Até 5	1
Médio	5 a 10	2
Complexo	Mais de 10	3

Tabela A1.8 - *Pesos dos Casos de Uso de acordo com o número de classes (NAGESWARAN, 2001)*

(iii) *Calcular o Pontos de Caso de Uso Não Ajustados*

O cálculo dos pontos de caso de uso não ajustados (PCUN) é realizado através da soma dos pesos dos atores (obtido em *(i)*) e pesos dos casos de uso (obtido em *(ii)*) do sistema. Assim:

$$PCUN = PA + PCU$$

(iv) *Calcular os Valores dos Fatores de Ajuste*

O método de ajuste dos pontos de caso de uso é bem similar ao adotado na técnica Análise de Pontos de Função, diferindo no fato de que, neste caso, são utilizados dois fatores de ajuste: *fator de complexidade técnica,* que considera as características técnicas do sistema, e *fator ambiental*, que considera as características ambientais que envolvem o sistema.

Cada fator de ajuste considera um série de caraterísticas que, ao serem analisadas, receberão um nível de influência que pode variar de 0 (zero) a 5 (cinco), onde 0 (zero) indica nenhuma influência, 1 (um) influência mínima, 2 (dois) influência moderada, 3 (três) influência média, 4 (quatro) influência significativa e 5 (cinco) grande influência. Cada nível de influência tem um valor numérico correspondente (peso) que será utilizado no cálculo dos fatores de ajuste.

O *fator de complexidade técnica* considera as seguintes características: Sistema Distribuído, Tempo de Resposta, Eficiência, Processamento Complexo, Código Reutilizável, Facilidade de Instalação, Facilidade de Uso, Portabilidade, Facilidade de Mudança, Concorrência, Recursos de Segurança, Acessibilidade por Terceiros e Necessidade de Treinamento Especial.

O *fator ambiental* considera as seguintes características: Familiaridade com RUP ou outro processo formal, Experiência com a aplicação em desenvolvimento,

Experiência em Orientação a Objeto, Presença de Analista Experiente, Motivação, Requisitos Estáveis, Desenvolvedores em Meio Expediente e Linguagem de Programação Complexa. Para as características do fator ambiental o nível de influência a elas atribuído corresponde ao nível de disponibilidade da característica em questão. Dessa forma, determinar que uma dada característica tem nível de influência alta (4 ou 5) significa que esta característica está presente no projeto e influencia no processo de desenvolvimento. Da mesma forma, atribuir um nível de influência baixo (0 ou 1) indica que a característica não está presente no processo de desenvolvimento.

Para calcular o *fator de complexidade técnica* utiliza-se a fórmula:

$$FCT = 0,6 + (0,01 \times TT)$$

Onde:

- FCT é o valor do fator de complexidade técnica
- TT é a soma dos pesos dos níveis de influência das características consideradas pelo fator de complexidade técnica atribuídos para o sistema.

Para calcular o *fator ambiental* utiliza-se a fórmula:

$$FA = 1,4 + (-0,03 \times TA)$$

Onde:

- FA é o valor do fator ambiental
- TA é a soma dos pesos dos níveis de influência das características consideradas pelo fator ambiental atribuídos para o sistema.

(v) *Calcular o Pontos de Caso de Uso Ajustados*

O cálculo dos pontos de caso de uso ajustados (PCUA) é realizado através da multiplicação dos pontos de casos de uso não ajustados (PCUNA), obtido em *(iii)*, pelos fatores de ajuste (FCT e FA), obtidos em *(iv)*. Assim:

$$PCUA = PCUN \times FCT \times FA$$

O número de pontos de caso de uso encontrado representa o tamanho do sistema. O cálculo das estimativas de prazo, esforço e custo é realizado da mesma forma que na Análise de Pontos de Função, ou seja, é necessário conhecer valores como o custo de um ponto de caso de uso e o tempo necessário para realizá-lo, ou o esforço para realizar um ponto de caso de uso e o custo do esforço. Com esses valores é possível calcular as estimativas para o projeto através das relações entre o número total de pontos de caso de uso do sistema e os valores de um ponto de caso de uso.

A1. 3 Exemplo de Contagem utilizando Análise de Pontos de Função[8]

Para exemplificar a utilização da técnica Análise de Pontos de Função, consideremos um pequeno sistema hipotético desenvolvido para uma academia de ginástica, com o objetivo de cadastrar os alunos matriculadose emitir um relatório gerencial que apresente o número de alunos matriculados totalizados por mês.

Considere o diagrama abaixo como representação do sistema hipotético. O arquivo (tabela) Alunos possui 10 atributos.

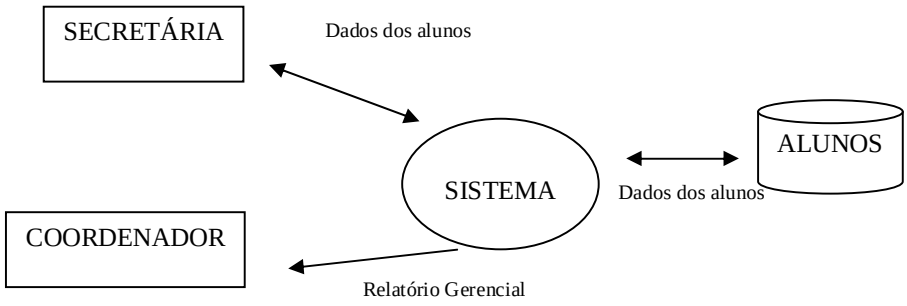


Figura A1.2 – Diagrama de Contexto do sistema hipotético.

Passos:

(i) *Determinar o Tipo de Contagem*

O tipo de contagem é para um *Projeto de Desenvolvimento*, uma vez que se trata de um sistema a ser desenvolvido e não de manutenção ou de medição em aplicação instalada.

[8] O exemplo aqui apresentado foi extraído de WEBER *et. al* (2001*)*

(ii) I*dentificar a Fronteira da Aplicação*

Não há interação com outros sistemas.

(iii) *Contar Funções Tipo Dados*

O número de Arquivos Lógicos Internos é 1 pois só há manipulação do arquivo Alunos. O número de Elementos de Dados é 10, que são os atributos do arquivo Alunos. Só há um Tipo de Registro em Alunos, pois não há especialização deste arquivo.

Não há Arquivos de Interface Externa, uma vez que não há interação com outros sistemas.

Arquivos Lógicos Internos	**Tipos de Elementos de Dados**	**Tipos de Elementos de Registros**
1 (Alunos)	10 (atributos de Alunos)	1
Arquivos de Interface Externa	**Tipos de Elementos de Dados**	**Tipos de Elementos de Registros**
0 (não há interação com outros sistemas)	0	0

Tabela A1.9 – Exemplo de Contagem de Funções Tipo Dados

(iv) *Contar Funções Tipo Transação*

Existem três Entradas Externas: inclusão, alteração e exclusão de alunos. Para as duas primeiras existem 10 Elementos de Dados, que são os atributos que são fornecidos como entrada para inclusão ou os atributos que podem ser modificados em uma alteração. Para a exclusão, apenas um Elemento de Dados é considerado, que é o código do aluno que será excluído. Em todas as entradas há apenas um arquivo referenciado: Alunos.

Supondo que o sistema possui uma consulta aos dados cadastrais, a função Consultas Externas apresenta contagem 1. Todos os atributos do arquivo Alunos são exibidos, totalizando 10 Elementos de Dados. Apenas o arquivo Alunos é utilizado, então Arquivos Referenciados é igual a 1.

A única Saída Externa é o Relatório Gerencial. Supondo que ele apresente: o código do aluno, nome do aluno, mês da matrícula, totalizador de alunos matriculados por mês e totalizador de alunos matriculados no ano, temos 5 Elementos de Dados. Apenas o arquivo Alunos é utilizado, então Arquivos Referenciados é igual a 1.

Entradas Externas	**Tipos de Elementos de Dados**	**Arquivos Referenciados**
1 (Inclusão)	10 (atributos de Alunos)	1 (Alunos)
1 (Alteração)	10 (atributos de Alunos)	1 (Alunos)
1 (Exclusão)	1 (código do alunos)	1 (Alunos)
Consultas Externas	**Tipos de Elementos de Dados**	**Arquivos Referenciados**
1 (Consulta aos dados cadastrais)	10 (atributos de Alunos)	1 (Alunos)
Saídas Externas	**Tipos de Elementos de Dados**	**Arquivos Reerenciados**
1 (Relatório Gerencial)	5 (informações apresentadas no relatório)	1 (Alunos)

Tabela A1.10 – Exemplo de Contagem de Funções Tipo Transação

(v) *Calcular os Pontos de Função Não Ajustados*

Analisando os valores obtidos no passos acima e as tabelas A1.1 a A1.4 chegamos aos seguintes valores:

Função	***Itens Contados por Complexidade***	***Contribuição***	***Total por Complexidade***	***Total de PFNA da Função***
ALI	1 Baixa 0 Média 0 Alta	x 7 x 10 x 15	7 0 0	7
AIE	0 Baixa 0 Média 0 Alta	x 5 x 7 x 10	0 0 0	0
EE	3 Baixa 0 Média 0 Alta	x 3 x 4 x 6	9 0 0	9
CE	1 Baixa 0 Média 0 Alta	x 3 x 4 x 6	3 0 0	3
SE	1 Baixa 0 Média 0 Alta	x 4 x 5 x 7	4 0 0	4

Total de Pontos de Função Não Ajustados: 23

Tabela A1.11 – Exemplo de cálculo dos pontos de função não ajustados

(v) *Calcular Valor do Fator de Ajuste*

Para calcular o fator de ajuste, as 14 características foram consideradas, obtendo-se os seguintes valores:

Características Gerais do Sistema	***Nível de Influência***	***Justificativa***
Comunicação de Dados	0	O sistema opera em micro *stand-alone*, portanto, não possui comunicação de dados.
Processamento Distribuído	0	O sistema opera em micro *stand-alone.*
Performance	1	Requisitos de *performance* foram estabelecidos, mas nenhuma ação especial foi necessária.
Configuração altamente utilizada	0	Não há restrições operacionais.
Volume de Transações	0	Nenhum período de pico de transações esperado.
Entrada de dados *on-line*	5	Sistema *on-line.*
Eficiência do usuário final	3	Sistema desenvolvido com interface gráfica.
Atualização *on-line*	3	Sistema *on-line*, sem proteção para perda de dados.
Processamento complexo	0	O sistema não executa processamento matemático ou de segurança.
Reusabilidade	1	O sistema foi desenvolvido levando-se em conta reuso de rotinas.
Facilidade de instalação	4	Utilização de ferramenta automática para implantação do sistema.
Facilidade de Operação	2	Sistema *on-line.*
Múltiplos locais	0	Nenhuma solicitação do usuário para implantar a aplicação em mais de um local..
Modificação facilitada	0	Nenhuma solicitação do usuário para projetar a aplicação visando minimizar ou facilitar mudanças.
Grau de Influência Total = 19		
VFA = (GIT * 0,01) + 0,65 = 0,84		

Tabela A1.12 – Exemplo de cálculo dos fator de ajuste

(vi) *Calcular Pontos de Função Ajustados*

Para ajustar os pontos de função do sistema, basta multiplicar os pontos de função não ajustados pelo valor do fator de ajuste, como apresentado abaixo:

PFA = 23 * 0,84 = 19,32

Sendo assim, o sistema hipotético possui 19 pontos de função.

Anexo 2

Realização de Estimativas utilizando COCOMOII

Este anexo apresenta os passos necessários para realizar estimativas utilizando COCOMO II e um exemplo de utilização do modelo.

A2.1 Os Modelos do COCOMO II

Conforme apresentado no capítulo 2, COCOMO II funciona como uma hierarquia de modelos que podem ser aplicados de acordo com a evolução do desenvolvimento do *software*. A definição dos modelos que o COCOMO II trata é apresentada a seguir (BOEHM *et al.*, 2000):

- *Composição da Aplicação:* este modelo é utilizado para realizar estimativas em projetos que utilizam ambientes ICASE *(Integrated Computer-Aided Software Engineering*) para apoiar as fases do desenvolvimento. Esses ambientes geralmente incluem as seguintes capacidades: um *framework* para aplicações com *middleware* para integrar e gerenciar a execução dos componentes da aplicação; um conjunto de utilitários comuns, tais como construtores de interfaces gráficas para o usuário, sistemas gerenciadores de bancos de dados e pacotes de suporte à rede; um conjunto de componentes reutilizáveis por domínio; um repositório capaz de gerenciar e permitir acesso a esses componentes; e ferramentas de desenvolvimento para projeto, implementação, integração e testes. É importante ressaltar que nem todas as aplicações são desenvolvidas dessa forma. Mas, para aplicações que utilizam ambientes ICASE a redução de prazos e esforços para desenvolver o produto é significativa. Devido a essas características a abordagem para realização das estimativas no modelo Composição da Aplicação é diferente da abordagem dos demais modelos. Esse modelo também pode ser utilizado na fase inicial do projeto, onde protótipos da interface com o usuário são construídos, por isso também é chamado Pré-Prototipação.

- *Pré-Projeto:* este modelo é utilizado para a realização de estimativas nas fases iniciais do projeto, onde as informações do sistema estão parcialmente definidas. Exemplo dessas informações pode ser a análise de requisitos do sistema, mesmo que ainda esteja incompleta.
- *Pós-Arquitetura:* este modelo é utilizado para a realização de estimativas do projeto quando este está pronto para ser desenvolvido, ou seja, quando toda a sua especificação, modelagem e projeto tenham sido realizados e informações detalhadas sobre o que se pretende construir já estejam definidas.

A2.2 Os Fatores de Equilíbrio do COCOMO II (BOEHM *et al.*, 2000)

Os fatores de equilíbrio são cinco características do projeto que contribuem para a determinação da relação entre o tamanho do sistema e o esforço necessário para desenvolvê-lo. Sua análise indicará o que ocorre com o esforço do projeto caso seu tamanho seja alterado. Cada fator de equilíbrio é definido por um conjunto de níveis de influência (muito baixo, baixo, médio, alto, muito alto e extremamente alto) e a cada nível de influência corresponde um valor numérico. Esses fatores só são considerados para os modelos Pré-Projeto e Pós-Arquitetura. São eles:

- **PREC - Precedência**: Indica o grau de similaridade do projeto em desenvolvimento com relação a outros desenvolvidos anteriormente.
- **FLEX - Flexibilidade no Desenvolvimento:** Indica o grau de flexibilidade do projeto no atendimento aos requisitos pré-estabelecidos.
- **RESL - Resoluções de Arquitetura e Risco:** Indica o grau de especificação realizada para o projeto no que diz respeito às definições de sua arquitetura e análise de riscos.
- **TEAM – Coesão da Equipe:** Indica o grau de interação das pessoas que compõem a equipe do projeto: usuários, clientes, desenvolvedores, analistas e outros.
- **PMAT – Maturidade do Processo:** Indica o grau de maturidade do processo em relação aos níveis de maturidade do CMM (*Capability Maturity Model*).

A2.3 Os Direcionadores de Custos do COCOMO II (BOEHM *et al.*, 2000)

Os direcionadores de custos são características do projeto que interferem diretamente no esforço necessário para seu desenvolvimento. Cada direcionador de custos é definido por um conjunto de níveis de influências (extremamente baixo, muito baixo, baixo, médio, alto, muito alto e extremamente alto) e a cada nível de influência corresponde um valor numérico, chamado multiplicador de esforço. Apenas os modelos Pré-Projeto e Pós-Arquitetura consideram os direcionadores de custos.

O modelo **Pós-Arquitetura** possui dezessete direcionadores de custos que são agrupados em quatro categorias: Atributos do Produto, Atributos da Plataforma, Atributos da Equipe de Desenvolvimento e Atributos do Projeto.

Atributos do Produto são as características do produto que influenciam na determinação do esforço necessário para desenvolvê-lo. Os direcionadores de custos que representam essas características são:

- **RELY - Confiabilidade Necessária ao *Software*:** Indica o efeito produzido pela execução incorreta das funções do *software*.
- **DATA - Tamanho da Base de Dados:** Indica o efeito provocado pelos requisitos de testes de dados no desenvolvimento do *software*. É medido através da relação entre o tamanho da base de dados de testes (em bytes) e o número de linhas de código do *software*. Essa relação é importante pois é a base para determinar o esforço necessário para inserir e manter os dados requeridos para realizar os testes do *software*.
- **CPLX - Complexidade do *Software*:** Indica a complexidade do *software* considerando cinco grupos de operações: operações de controle, operações computacionais, operações dependentes de dispositivos, operações de administração de dados e operações de gerência de interface com o usuário.
- **RUSE - Reutilizabilidade Necessária:** Indica o grau de reutilização dos componentes que serão desenvolvidos durante a construção do produto.
- **DOCU - Documentação Compatível com as Necessidades do Ciclo de Vida:** Indica o grau de documentação disponível para atender às necessidades do ciclo de vida do *software*.

Atributos da Plataforma referem-se às características que devem ser observadas baseando-se nas estruturas de *hardware* e *software* a serem utilizadas. Os direcionadores de custos que representam essas características são:

- **TIME - Restrições de Tempo de Execução:** Indica as restrições de tempo impostas para o *software*. Seu valor é expresso pelo percentual do tempo de execução disponível que será consumido para a execução das operações.
- **STOR - Restrições da Memória Principal:** Indica as restrições impostas para o armazenamento dos dados na memória principal do sistema. Seu valor é expresso pelo percentual da capacidade de armazenamento disponível que será consumida para a operação do *software*.
- **PVOL - Mudanças de Plataforma:** Indica o tempo previsto para que mudanças na plataforma sejam necessárias.

Atributos da Equipe de Desenvolvimento referem-se às características que devem ser observadas na equipe que irá desenvolver o *software*. Depois do tamanho do produto, considera-se que essas são as características que mais fortemente influenciam na determinação do esforço necessário para o desenvolvimento . Os direcionadores de custos que representam essas características são:

- **ACAP – Capacidade dos Analistas:** Indica a capacidade dos analistas como equipe e não individualmente. Considera habilidades de análise, projeto, comunicação, cooperação e eficiência.
- **PCAP – Capacidade dos Programadores:** Indica a capacidade dos programadores como uma equipe e não individualmente. Considera habilidades lógicas, de comunicação, cooperação e eficiência.
- **PCON – Rotatividade de Pessoal:** Indica a taxa de rotatividade anual de pessoal.
- **APEX – Experiência na Aplicação:** Indica o tempo de experiência da equipe no desenvolvimento de aplicações similares ao *software* a ser desenvolvido.
- **PLEX – Experiência com a Plataforma:** Indica o tempo de experiência da equipe na utilização da plataforma de desenvolvimento utilizada no *software*.

- **LTEX – Experiência com a Linguagem e Ferramentas:** Indica o tempo de experiência da equipe na utilização das ferramentas e linguagens utilizadas no *software* a ser desenvolvido.

Atributos do Projeto referem-se às características que devem ser observadas no projeto a ser desenvolvido, tais como: utilização de tecnologias, localização da equipe de desenvolvimento e restrições de cronograma. Os direcionadores de custos que representam essas características são:

- **TOOL - Uso de Ferramentas de *Software*:** Indica o grau de utilização de ferramentas de *software* para apoiar o desenvolvimento do produto.
- **SITE - Desenvolvimento Multilocal:** Indica a localização da equipe e a forma de comunicação entre seus membros.
- **SCED - Prazo Necessário para o Desenvolvimento:** Indica as restrições impostas no cronograma de desenvolvimento do projeto. Sua definição se dá através da porcentagem do prazo previsto que representa a restrição de tempo, ou seja, o tempo em que o sistema deverá ser desenvolvido.

O modelo **Pré-Projeto** possui sete direcionadores de custos. A redução do número de direcionadores se dá pois nesse modelo muitas informações ainda não foram definidas, o que dificulta o detalhamento de todos os direcionadores definidos anteriormente. Alguns direcionadores desse modelo correspondem a uma combinação das características tratadas nos direcionadores apresentados anteriormente. Os direcionadores do modelo Pré-Projeto são:

- **RCPX - Confiabilidade e Complexidade do Produto:** Indica o grau de confiabilidade e complexidade do produto. É a combinação dos direcionadores RELY, DATA, CPLX e DOCU.
- **RUSE – Reutilizabilidade Necessária:** É o mesmo direcionador utilizado no modelo Pós-Arquitetura. Indica o grau de reutilização dos componentes que serão desenvolvidos durante a construção do produto.
- **PDIF – Dificuldades com a Plataforma:** Indica o grau de dificuldade com a plataforma de desenvolvimento levando-se em consideração restrições de tempo,

memória e características da plataforma. É a combinação dos direcionadores TIME, STOR e PVOL.

- **PERS - Capacidade do Pessoal:** Indica a capacidade do pessoal envolvido no desenvolvimento do produto, considerando analistas e programadores. É a combinação dos direcionadores ACAP, PCAP e PCON.
- **PREX – Experiência Profissional do Pessoal:** Indica o grau de experiência do pessoal envolvido no desenvolvimento do produto no que diz respeito à plataforma, linguagem e aplicação. É a combinação dos direcionadores APEX, PLEX e LTEX.
- **FCIL – Facilidades:** Indica o grau de utilização de facilidades para o desenvolvimento do *software*, como por exemplo a utilização de ferramentas de apoio ao desenvolvimento do *software* e ao desenvolvimento multilocal. É a combinação dos direcionadores TOOL e SITE.
- **SCED - Prazo Necessário para o Desenvolvimento:** É o mesmo direcionador do modelo Pós-Arquitetura. Indica as restrições impostas no cronograma de desenvolvimento do projeto. Sua definição se dá através da porcentagem do prazo previsto que representa a restrição de tempo, ou seja, o tempo em que o sistema deverá ser desenvolvido.

A2.4 O Processo de Realização de Estimativas utilizando COCOMO II (BOEHM *et al., 2000)*

O processo de realização de estimativas de *software* utilizando o COCOMO II pode ser resumido em quatro etapas: *(i)* determinar o modelo da aplicação; *(ii)* realizar estimativas de esforço; *(iii)* realizar estimativas de prazo; e *(iv)* realizar estimativas de custos. Essas etapas são descritas a seguir:

(i) *Determinar o Modelo da Aplicação*

Para realizar estimativas utilizando COCOMO II, inicialmente é necessário analisar qual será o modelo utilizado, de acordo com as características da fase de desenvolvimento em que o *software* se encontra. Os modelos do COCOMO II foram apresentados na seção A2.1.

(ii) *Realizar Estimativas de Esforço*

Após ter sido identificado o modelo pertinente ao projeto, deve-se realizar a estimativa de esforço.

Os modelos Pré-Projeto e Pós-Arquitetura seguem os mesmos passos para a realização da estimativa de esforço, porém, o modelo Composição da Aplicação segue um caminho diferente, devido a suas particularidades anteriormente mencionadas.

O item *(ii.a)* apresenta a descrição dos passos utilizados para o cálculo do esforço nos modelos Pré-Projeto e Pós-Arquitetura. O item *(ii.b)* descreve os passos para calcular a estimativa de esforço no modelo Composição da Aplicação. Em ambos, o resultado é apresentado na unidade pessoas/mês, que indica quantas pessoas são necessárias para completar o desenvolvimento do *software* em um mês.

(ii.a) *Cálculo do Esforço para os modelos Pré-Projeto e Pós-Arquitetura*

Para realizar a estimativa de esforço é utilizada a fórmula a seguir:

$$E_{(P/M)} = A \times tamanho^{G} \times \sum_{i=1}^{i=n} ME_i$$

Onde:

- **A** é uma constante única para todos os modelos, com valor 2,94. Essa constante, entretanto, pode ser alterada se novas calibrações forem realizadas[9].
- O **tamanho** do *software* pode ser definido através do número de linhas de código fonte ou através do número de pontos de função. Quando o tamanho do *software* é definido em número de pontos de função, o mesmo deve ser convertido para número de linhas de código fonte através da utilização de

[9] Conforme apresentado no capítulo 2, novas calibrações podem ser realizadas pela equipe que propôs o COCOMO II, por outros grupos de pesquisa ou, ainda, desenvolvida por empresas que utilizem o modelo com base em suas próprias experiências no desenvolvimento de *software*. Essas calibrações são realizadas para melhorar a adequação do COCOMO II aos projetos/empresa onde é aplicado, tornando as estimativa mais acuradas. As calibrações também podem ser aplicadas a outras constantes e aos valores dos fatores de equilíbrio e direcionadores de custos.

tabelas padrão que indicam quantas linhas de código fonte correspondem a um ponto de função em uma determinada linguagem.

- Para determinar o valor de **G**[10], utiliza-se a fórmula $G = B + 0{,}01 \times \sum_{i=1}^{i=5} FE_i$,

  onde **B** é uma constante, com valor 0,91. Essa constante, entretanto, pode ser alterada se novas calibrações forem realizadas.

- O valor de $\sum_{i=1}^{i=5} FE_i$ é obtido através da análise dos cinco fatores de equilíbrio (FE), apresentados em A2.2

  Para cada fator de equilíbrio deve ser indicado um valor relacionado ao nível de influência deste para o projeto (muito baixo, baixo, médio, alto, muito alto e extremamente alto). Ao nível indicado corresponderá um valor numérico a ser considerado, de acordo com as tabelas A2.1 e A2.2, apresentadas a seguir.

[10] Analisando o valor do expoente **G** é possível conhecer o comportamento do projeto em situações de aumento do tamanho do mesmo. Se G < 1 o projeto em questão exibe o que BOEHM *et. al* (2000) chama de escala econômica, ou seja, se o tamanho do projeto dobrar , por exemplo, o esforço necessário para desenvolvê-lo será menor que o dobro do calculado. Se G = 1, as alterações de tamanho e esforço são equivalentes. Se G > 1 o projeto exibe uma escala decrescente de economia, ou seja, se o tamanho do sistema aumentar, o esforço necessário para desenvolvê-lo aumentará em proporções maiores.

Fatores	**Níveis de Inluência**					
	Muito baixo	*Baixo*	*Médio*	*Alto*	*Muito alto*	*Extremamente alto*
PREC	Nenhuma precedência	Pouca precedência	Precedência moderada	Alguma familiaridade	Muita familiaridade	Completamente familiar
FLEX	Muito Rigoroso	Flexibilidade ocasional	Flexibilidade moderada	Conformidade usual	Alguma conformidade	Metas usuais
RESL	Pequeno (20%)	Algum (40%)	Regular (60%)	Geral (75%)	Bom (90%)	Completo (100%)
TEAM	Interações muito difíceis	Alguma dificuldade de interação	Interações basicamente cooperativas	Amplamente cooperativo	Altamente cooperativo	Interação total
PMAT	Inferior ao CMM nível 1	Superior ao CMM nível 1	CMM nível 2	CMM nível 3	CMM nível 4	CMM nível 5

Tabela A2.1 - Determinação do Nível de Influência dos Fatores de Equilíbrio

Fatores de Equilíbrio	**Nível de Influência**					
	Muito baixo	*Baixo*	*Médio*	*Alto*	*Muito alto*	*Extremamente alto*
PREC	6,20	4,96	3,72	2,48	1,24	0,00
FLEX	5,07	4,05	3,04	2,03	1,01	0,00
RESL	7,07	5,65	4,24	2,83	1,41	0,00
TEAM	5,48	4,38	3,29	2,19	1,10	0,00
PMAT	7,80	6,24	4,68	3,12	1,56	0,00

Tabela A2.2 – Valores atribuídos aos Fatores de Equilíbrio de acordo com os Níveis de Influência

- O valor de $\sum_{i=1}^{i=n} ME_i$ na fórmula de esforço é obtido através do somatório dos valores resultantes da análise dos níveis de influência dos direcionadores de custos de cada modelo, que foram apresentados na seção A2.3. Neste momento o direcionador de custos SCED não deve ser considerado.

  Para cada direcionador de custos deve ser indicado um valor relacionado ao nível de influência deste para o projeto (extremamente baixo, muito baixo, baixo, médio, alto, muito alto e extremamente alto). Ao nível indicado corresponderá um valor numérico (multiplicador de esforço – ME) a ser considerado, de acordo com as tabelas A2.3 a A2.5, apresentadas a seguir:

Direcionador	**Nível de Influência**					
	Muito baixo	*Baixo*	*Médio*	*Alto*	*Muito alto*	*Extremamente alto*
RELY	Leve inconveniência	Perdas facilmente recuperadas	Perdas moderadas	Altas perdas financeiras	Risco à vida humana	
DATA		Tamanho do BD de testes/SLOC do programa < 10	10 ≤ Tamanho do BD de testes/SLOC do programa < 100	100 ≤ Tamanho do BD de testes/SLOC do programa < 1K	Tamanho do BD de testes/SLOC do programa ≥ 1K	
CPLX[11]						
Operações de Controle	Código simples, com poucos operadores sendo usados em programação não estruturada. Composição simples do módulo, via chamadas a procedimentos ou scripts simples.	Operadores em uso apenas em programação estruturada. Predicados simples.	Uso de tabelas de decisão. Algum suporte a processamento distribuído Aninhamentos simples.	Considerável controle entre módulos. Controle de filas e pilhas. Operadores aninhados. Processamentos distribuídos homogêneos. Controle simples de processador de tempo real.	Código reentrante e recursivo. Manipulação de interrupções. Sincronização de tarefas. Chamadas complexas. Processamentos Distribuídos Heterogêneos. Controle de processador de tempo real.	Controle de microcódigo. Programação dinâmica de recursos. Controle distribuído em tempo real .
Operações computacionais	Avaliação de expressões simples.	Avaliação de expressões moderadas.	Uso de rotinas matemáticas e estatísticas padrão. Operações básicas de matrizes e vetores.	Análise numérica básica, interpolação multivariada, equações diferenciais ordinárias.	Análise numérica complexa. Equações matriciais. Equações diferenciais parciais. Paralelismo simples.	Análise numérica difícil e não estruturada, dados estocásticos. Paralelismo complexo.

Tabela A2.3 - Determinação dos Níveis de Influência dos Direcionadores de Custos do Modelo Pós-Arquitetura

[11] O gerente deve analisar todas as áreas do direcionador que se aplicam ao produto a ser desenvolvido e determinar seu nível de influência através de uma média subjetiva dos níveis de influência de suas áreas. O mesmo ocorre para o direcionador SITE.

Direcionador	**Nível de Influência**					
	Muito baixo	*Baixo*	*Médio*	*Alto*	*Muito alto*	*Extremamente alto*
CPLX						
Operações dependentes de dispositivos	Instruções de leitura e gravação simples, com formatos simples.	Nenhum conhecimento necessário sobre as características do processador ou dispositivos de entrada e saída. As entradas e saídas são executadas no nível GET/PUT.	Processamento de entrada e saída inclui seleção de dispositivo, checagem de status e processamento de erros.	Operações em nível físico de entrada e saída. Otimização de entradas e saídas duplicadas.	Rotinas para diagnóstico de interrupção. Manipulação da linha de comunicação. Sistemas embutidos de processamento intenso.	Operações microprogramadas. Sistemas embutidos de processa-mento crítico.
Operações de administração de dados	Arrays simples na memória principal, consultas e atualizações simples.	Arquivos simples, sem edições, alterações de estruturas de dados ou arquivos intermediá-rios. Consultas e atualizações moderada-mente complexas.	Entrada multiarquivo e saída em monoarquivo. Mudanças estruturais simples. Consultas e atualizações complexas.	Triggers simples ativadas pelo fluxo de dados. Reestruturação de dados complexa.	Coordenação de bases de dados distribuídas. Triggers complexas. Otimização de pesquisas.	União de estruturas relacionais e por objetos. Gerência de dados em linguagem natural.
Operações de gerência de interface com usuário	Formulários simples e geradores de relatórios.	Uso de construtores simples de interface gráfica.	Uso simples de conjunto widget	Desenvolvimento e extensão de conjuntos widget. Recursos simples de entrada e saída de voz e multimídia.	Recursos moderada-mente complexos de 2D/3D, gráficos dinâmicos e multimídia.	Recursos multimídia complexos. Realidade virtual. Interface em linguagem natural.
RUSE		Nenhum	Aproveitamento em projetos.	Aproveitamento em programas.	Aproveitamento em linhas de produtos.	Aproveitamento em múltiplas linhas de produtos.

Tabela A2.3 - Determinação dos Níveis de Influência dos Direcionadores de Custos do Modelo Pós-Arquitetura (continuação)

Direcionador	**Nível de Influência**					
	Muito baixo	*Baixo*	*Médio*	*Alto*	*Muito alto*	*Extremamente alto*
DOCU	Muito escassa para as necessidades Do ciclo de vida.	Pouca para as necessidades do ciclo de vida.	Adequada para as necessidades do ciclo de vida.	Excessiva para as necessidades do ciclo de vida.	Muito excessiva para as necessidades do ciclo de vida.	
TIME			Utilização ≤ 50% do tempo de execução disponível	Utilização de 70% do tempo de execução disponível	Utilização de 85% do tempo de execução disponível	Utilização de 95% do tempo de execução disponível
STOR			Utilização ≤ 50% do armazena-mento disponível	Utilização de 70% do armazena-mento disponível	Utilização de 85% do armazena-mento disponível	Utilização de 95% do armazenamento disponível
PVOL		Período para grande mudanças: 12 meses Pequenas: 1 mês	Período para grande mudanças: 6 meses Pequenas: 2 semanas	Período para grande mudanças: 2 meses Pequenas: 1 semana	Período para grande mudanças: 2 semanas Pequenas: 2 dias	
ACAP	15º percentil	35º percentil	55º percentil	75º percentil	90º percentil	
PCAP	15º percentil	35º percentil	55º percentil	75º percentil	90º percentil	
PCON	Rotatividade de pessoal de 48% ao ano	Rotatividade de pessoal de 24% ao ano	Rotatividade de pessoal de 12% ao ano	Rotatividade de pessoal de 6% ao ano	Rotatividade de pessoal de 3% ao ano	
APEX	≤ 2 meses de experiência	6 meses de experiência	1 ano de experiência	3 anos de experiência	6 anos de experiência	
PLEX	≤ 2 meses de experiência	6 meses de experiência	1 ano de experiência	3 anos de experiência	6 anos de experiência	

Tabela A2.3 - Determinação dos Níveis de Influência dos Direcionadores de Custos do Modelo Pós-Arquitetura (continuação)

Direcionador	**Nível de Influência**					
	Muito baixo	*Baixo*	*Médio*	*Alto*	*Muito alto*	*Extremamente alto*
LTEX	≤ 2 meses de experiência	6 meses de experiência	1 ano de experiência	3 anos de experiência	6 anos de experiência	
TOOL	Uso de ferramentas para edição, codificação e depuração.	Uso de ferramentas CASE simples. Pequena integração.	Uso de ferramentas básicas, moderada-mente integradas.	Uso de ferramentas pesadas, moderada-mente integradas.	Uso de ferramentas pesadas, pró-ativas, bem integradas, com processos e métodos. Reutilização.	
SITE						
Localização	Internacional	Múltiplas cidades **e** empresas.	Múltiplas cidades **ou** empresas.	Mesma cidade ou área metropolitana.	Mesmo prédio ou conjunto de prédios.	Desenvolvimento em um único local..
Comunicações	Telefone e correio.	Telefone individual e fax.	Correio eletrônico em banda estreita.	Comunicação eletrônica em banda larga.	Comunicação eletrônica em banda larga e ocasionalmente vídeo conferência.	Multimídia interativa.
SCED	Execução do projeto em 75% do cronograma original	Execução do projeto em 85% do cronograma original	Execução do projeto em 100% do cronograma original	Execução do projeto em 130% do cronograma original	Execução do projeto em 160% do cronograma original	

Tabela A2.3 - Determinação dos Níveis de Influência dos Direcionadores de Custos do Modelo Pós-Arquitetura (continuação)

Direcionador	**Níveis de Influência**						
	Extremamente baixo	***Muito baixo***	***Baixo***	***Médio***	***Alto***	***Muito alto***	***Extremamente Alto***
RCPX[12]							
Soma dos níveis de influência de RELY, DATA, CPLX e DOCU	5, 6	7, 8	9-11	12	13-15	16-18	19-21
Combinação de confiança (RELY) e documentação (DOCU)	Muito pequena	Pequena	Alguma	Básica	Forte	Muito forte	Extremamente forte
Complexidade do produto (CPLX)	Muito simples	Simples	Alguma	Moderada	Complexo	Muito complexo	Extremamente complexo
Tamanho da Base de Dados (DATA)	Pequena	Pequena	Pequena	Moderada	Grande	Muito grande	Muito grande
RUSE	Idem tabela A2.3						
PDIF							
Soma dos níveis de influência de TIME, STOR e PVOL			8	9	10-12	13-15	16, 17

Tabela A2.4 - Determinação dos Níveis de Influência dos Direcionadores de Custos do Modelo Pré-Projeto

[12] O gerente deve analisar todos os itens do direcionador e determinar seu nível de influência através de uma média subjetiva dos níveis de influência dos itens. Para realizar a soma dos níveis dos direcionadores do modelo Pós-Arquitetura que compoem o direcionador do modelo Pré-Projeto (primeiro item) basta considerar que o nível de influência 'muito baixo' para o direcionador no modelo Pós-Arquitetura corresponde ao valor 1, 'baixo' corresponde ao valor 2, 'médio' ao valor 3, 'alto' ao valor 4, 'muito alto' ao valor 5 e 'extremamente alto' ao valor 6. Essas observações valem também para os direcionadores PDIF, PERS, PREX e FCIL.

Direcionador	**Níveis de Influência**						
	Extremamente baixo	***Muito baixo***	***Baixo***	***Médio***	***Alto***	***Muito alto***	***Extremamente Alto***
PDIF							
Mudanças de plataforma			Muito estável	Estável	Alguma volatilidade	Volátil	Muito volátil
Combinação das restrições de tempo (TIME) e memória (STOR)			≤ 50%	≤ 50%	65%	80%	90%
PERS							
Soma dos níveis de influência de ACAP, PCAP e PCON	3, 4	5, 6	7, 8	9	10, 11	12,13	14, 15
Combinação da capacidade dos analistas (ACAP) e programadores (PCAP)	20%	35%	45%	55%	65%	75%	85%
Rotatividade de pessoal (PCON)	45%	30%	20%	12%	9%	6%	4%
PREX							
Soma dos níveis de influência de APEX, PLEX e LTEX	3, 4	5, 6	7, 8	9	10, 11	12, 13	14, 15

Tabela A2.4 - Determinação dos Níveis de Influência dos Direcionadores de Custos do Modelo Pré-Projeto (continuação)

Direcionador	**Níveis de Influência**						
	Extremamente baixo	***Muito baixo***	***Baixo***	***Médio***	***Alto***	***Muito alto***	***Extremamente Alto***
PREX							
Combinação de experiências com a aplicação (APEX), plataforma (PLEX) e linguagem (LTEX)	≥ 3 meses	5 meses	9 meses	1 ano	2 anos	4 anos	6 anos
FCIL							
Soma dos níveis de influência de TOOL e SITE	2	3	4, 5	6	7, 8	9, 10	11
Facilidades oferecidas por ferramentas (TOOL)	Mínimo	Algum	Uso de ferramentas CASE simples.	Uso de ferramentas CASE básicas.	Uso de boas ferramentas, moderadamente integradas.	Uso de ferramentas muito boas, moderadamente integradas.	Uso de ferramentas muito boas e bem integradas.
Apoio para desenvolvimento multilocal (SITE)	Pouco apoio para desenvolvi-mentos multilocal complexos.	Algum apoio para desenvolvi-mentos multilocal complexos.	Algum apoio para desenvolvi-mentos multilocal moderadamente complexos.	Apoio básico para desenvolvi-mentos multilocal moderadamente complexos.	Apoio forte para desenvolvi-mentos multilocal moderadamente complexos.	Apoio forte para desenvolvi-mentos multilocal simples.	Apoio muito forte para desenvolvi-mentos multilocal simples ou desenvolvi-mentos em um único. Local.
SCED	Idem tabela A2.3						

Tabela A2.4 - Determinação dos Níveis de Influência dos Direcionadores de Custos do Modelo Pré-Projeto (continuação)

Direcionadores	**Nível de Influência**						
	Extrema-Mente baixo	*Muito baixo*	*Baixo*	*Médio*	*Alto*	*Muito alto*	*Extrema-mente alto*
RELY		0,82	0,92	1,00	1,10	1,26	
DATA			0,90	1,00	1,14	1,28	
CPLX		0,73	0,87	1,00	1,17	1,34	1,74
RUSE			0,95	1,00	1,07	1,15	1,24
DOCU		0,81	0,91	1,00	1,11	1,23	
TIME				1,00	1,11	1,29	1,63
STOR				1,00	1,05	1,17	1,46
PVOL			0,87	1,00	1,15	1,30	
ACAP		1,42	1,19	1,00	0,85	0,71	
PCAP		1,34	1,15	1,00	0,88	0,76	
PCON		1,29	1,12	1,00	0,90	0,81	
APEX		1,22	1,10	1,00	0,88	0,81	
PLEX		1,19	1,09	1,00	0,91	0,85	
LTEX		1,20	1,09	1,00	0,91	0,84	
TOOL		1,17	1,09	1,00	0,90	0,78	
SITE		1,22	1,09	1,00	0,93	0,86	0,80
SCED		1,43	1,14	1,00	1,00	1,00	
RCPX	0,49	0,60	0,83	1,00	1,33	1,91	2,72
PDIF			0,87	1,00	1,29	1,81	2,61
PERS	2,12	1,62	1,26	1,00	0,83	0,63	0,50
PREX	1,59	1,33	1,12	1,00	0,87	0,74	0,62
FCIL	1,43	1,30	1,10	1,00	0,87	0,73	0,62

Tabela A2.5 - Determinação dos Multiplicadores de Esforço dos Níveis de Influência dos Direcionadores de Custos

Para processos de manutenção devem ser utilizados os direcionadores de custos do modelo Pós-Arquitetura, excluindo-se RUSE e SCED.

(ii.b) *Cálculo do Esforço para os modelos Composição da Aplicação*

O modelo Composição da Aplicação, como mencionado anteriormente, tem características diferentes dos demais modelos, por isso o cálculo do esforço estimado necessário para executá-lo ocorre de outra forma.

Utilizar número de linhas de código fonte para estimar o tamanho do *software* nesse modelo pode ser inexpressivo, uma vez que quando há utilização de geradores de interface, linguagens visuais e outras facilidades dos ambientes ICASE, o número de linhas de código não é capaz de refletir o tamanho real do produto. O mesmo ocorre para os pontos de função. Sendo assim, os pesquisadores do COCOMO II (BOEHM *et al., 2000)* acrescentaram escalas de produtividade à técnica de medição em pontos de objeto e a chamaram de medição em pontos de aplicação. Essa medição estima o tamanho do sistema através de estimativas do número de telas, relatórios e componentes de linguagem de 3ª geração (3GL). A maioria dos autores consideram essa medição ainda com o nome original (pontos de objeto).

Para estimar o esforço nesse modelo, sete passos devem ser seguidos:

Passo 1: **Estimar o número de telas, relatórios e componentes de linguagem de 3ª geração do *software*.**

Neste passo são contados os elementos presentes no *software*. As definições padrão do ambiente ICASE também devem ser consideradas.

Passo 2: **Classificar cada elemento identificado segundo seu nível de complexidade.**

Para classificar os elementos é preciso consultar as tabelas A2.6, para classificar as telas, e A2.7, para classificar os relatórios. Nas tabelas apresentadas a seguir, ***srv*** é o número de tabelas de dados do servidor utilizadas na tela ou relatório e ***cln*** é o número de tabelas de dados de clientes utilizadas na tela ou relatório.

Número de Views	*Fontes das tabelas de dados*		
	Total < 4 ***(< 2 srv, < 3 cln)***	***Total < 8*** ***(2-3 srv, 3-5 cln)***	***Total ≥ 8*** ***(> 3 srv, > 5 cln)***
> 3	Simples	Simples	Média
3 – 7	Simples	Média	Difícil
≥ 8	Média	Difícil	Difícil

Tabela A2.6 – Níveis de Complexidade de Telas

Número de Seções	*Fontes das tabelas de dados*		
	Total < 4 ***(< 2 srv, < 3 cln)***	***Total < 8*** ***(2-3 srv, 3-5 cln)***	***Total ≥8*** ***(> 3 srv, > 5 cln)***
0 – 1	Simples	Simples	Média
2 – 3	Simples	Média	Alta
>3	Média	Alta	Alta

Tabela A2.7 – Níveis de Complexidade de Relatórios

Passo 3: **Avaliar o peso correspondente à complexidade determinada para os elementos identificados.**

Esse passo é executado consultando-se a tabela a seguir:

Tipo de Elemento	*Pesos*		
	Complexidade Simples	***Complexidade Média***	***Complexidade Alta***
Telas	1	2	3
Relatórios	2	5	8
Componentes 3GL	-	-	10

Tabela A2.8 – Pesos das Complexidade dos Elementos

Passo 4: **Determinar o número de pontos de aplicação do *software*.**

Esse passo é executado através da fórmula a seguir:

$$PA = \sum T + \sum R + \sum C$$

Onde:

PA = número de pontos de aplicação

$\sum T$ = (nº de telas com complexidade simples) +
(nº de telas com complexidade média x 2) +
(nº de telas com complexidade alta x 3)

$\sum R$ = (nº de relatórios com complexidade simples x 2) +
(nº de relatórios com complexidade média x 5) +
(nº de relatórios com complexidade alta x 8)

$\sum C$ = (nº de componentes 3GL x 10)

Passo 5: **Estimar a porcentagem de reutilização esperada para o projeto.**

Corresponde à porcentagem de telas, relatórios e componentes de linguagens de 3ª geração que serão utilizadas de aplicações desenvolvidas anteriormente. Após esse valor ser estimado, o número de novos pontos de aplicação deve ser calculado, através da fórmula:

$$NPA = PA \times (100 - \%R) / 100$$

Onde:

NPA = número de pontos de aplicação do *software* considerando a reutilização de componentes de outras aplicações

PA = número de pontos de aplicação sem considerar a reutilização de componentes (valor estimado no passo 4)

%R = porcentagem de reutilização estimada

Passo 6: **Determinar a taxa de produtividade.**

Esse passo é executado consultando-se a tabela a seguir:

Característica	***Níveis***				
Experiência e capacidade dos desenvolvedores	*Muito baixo*	*Baixo*	*Médio*	*Alto*	*Muito alto*
Maturidade e Capacidade do ambiente ICASE	*Muito baixo*	*Baixo*	*Médio*	*Alto*	*Muito alto*
PROD (taxa de produtividade)	4	7	13	25	50

Tabela A2.9 – Taxa de Produtividade

Passo 7: **Realizar a estimativa de esforço**.

Esse passo é executado através da fórmula a seguir:

$$E_{(PM)} = NAP/PROD$$

Onde:

E = esforço estimado (pessoas/mês)

NAP = número dos novos pontos de aplicação (estimado no passo 5)

PROD = taxa de produtividade (estimada no passo 6)

(iii) *Realizar Estimativas de Prazo*

Após ter sido estimado o esforço (itens *(ii.a)* ou *(ii.b)*) o mesmo deve ser feito para o prazo.

Para calcular o prazo estimado para o projeto, inicialmente deve ser feito o cálculo do prazo original, ou seja, sem considerar qualquer restrição do tempo de desenvolvimento para o projeto. Essa estimativa é realizada através da fórmula

$$P_{(M)} = C \times E^F$$

que fornece o prazo em meses, onde:

- **C** é uma constante única para todos os modelos, com valor 3,67. Essa constante, entretanto, pode ser alterada se novas calibrações forem realizadas.
- $F = D + 0{,}2 \times (G - B)$, onde $D = 0{,}28$ (podendo ser alterado por calibrações) e **G** é o valor encontrado no cálculo do esforço.
- **E** é o valor do esforço resultante da etapa *(ii)* do processo.

Obtido o prazo original previsto para a realização do projeto é preciso observar se há restrição de tempo de desenvolvimento para o mesmo e qual a relação entre o prazo original previsto e essa restrição, caso exista. Essa relação é considerada pelo direcionar SCED que indica o grau de compressão do prazo original para atender às restrições de tempo especificadas para o projeto. Para incluir o direcionador SCED no cálculo do esforço, basta multiplicar o prazo original ($P_{(M)} = C \times E^F$) por $SCED\% / 100$, onde:

- **SCED%** indica a porcentagem de compressão correspondente ao multiplicador de esforço atribuído ao direcionador de custos SCED. Por exemplo: durante a análise do prazo originalmente previsto, o gerente do projeto observou que o projeto deveria ser concluído em 70% do prazo originalmente previsto, sendo assim, atribuiu ao direcionador SCED o nível de influência ‘muito baixo’, resultando no multiplicador de esforço 1,43 (obtido analisando-se as tabelas A2.3 e A2.5). Para realizar a estimativa do prazo considerando a restrição e tempo, é preciso analisar qual a porcentagem de compressão do direcionar de custos SCED no nível de influência correspondente ao multiplicador 1,43, isto é, nível ‘muito baixo’. Observando a tabela A2.3 é possível notar que o nível ‘muito baixo’

corresponde a uma compressão de 75% tempo original. Sendo assim, SCED% receberia o valor 75.

(iv) *Realizar Estimativas de Custos*

A estimativa dos custos depende dos resultados obtidos nas etapas *(ii)* e *(iii).*

Para obter o número médio de pessoas que deverão ser alocadas ao projeto, basta dividir o valor do esforço pelo prazo. O COCOMO II considera que um mês equivale a 152 horas de trabalho, mas este valor pode ser modificado para se adequar a empresas específicas.

Para proceder no cálculo dos custos, é necessário conhecer o custo dos recursos humanos que serão utilizados no projeto, como por exemplo, o valor/hora ou valor/mês de um profissional ou o valor/hora padrão utilizado pela organização, definido a partir da análise de dados de projetos realizados anteriormente.

Para realizar a estimativa dos custos, basta calcular o produto do custo de uma unidade de esforço (valor/hora por exemplo) pelo tempo de utilização do mesmo.

Por exemplo, se as estimativas de esforço e prazo de um projeto foram respectivamente 5 pessoas e 10 meses, para estimar os custos desse projeto basta multiplicar o valor/ mês de cada pessoa pelo prazo em que elas estarão no projeto, que, nesse caso, são 10 meses.

A2. 5 Exemplo de estimativas utilizando COCOMO II

Para exemplificar a utilização do COCOMO II, consideremos o sistema hipotético desenvolvido para uma academia de ginástica do exemplo de contagem utilizando Análise de Pontos de Função apresentado no Anexo 1.

(i) *Determinar o Modelo da Aplicação*

Suponhamos que uma primeira versão da especificação de requisitos do sistema já tenha sido realizada, porém os detalhes ainda não foram descritos. Sendo assim, o modelo utilizado será *Pré-Projeto*.

(ii) *Realizar Estimativas de Esforço*

Para realizar o cálculo da estimativa de esforço para o sistema, os fatores de equilíbrio e direcionadores de custos devem ser analisados. O gerente do projeto realizou essa análise e forneceu os seguintes valores:

Fatores de Equilíbrio	***Nível de Influência***	***Justificativa***	***Valor numérico***
PREC – Precedência	Muito alto	Muita Familiaridade	1,24
FLEX – Flexibilidade no Desenvolvimento	Extremamente alto	Metas usuais	0,00
RESL – Resoluções de Arquitetura e Risco	Muito alto	Bom (90%)	1,41
TEAM – Coesão da equipe	Muito alto	Altamente cooperativo	1,10
PMAT – Maturidade do processo	Médio	CMM nível 2	4,68

Tabela A2.10 – Valores dos fatores de equilíbrio para o exemplo apresentado

Direcionadores de Custos	***Nível de Influência***	***Justificativa***	***Multiplicador de Esforço***
RCPX – Confiabilidade e Complexidade do Produto	Baixo	Confiança e documentação básica, pouca complexidade e base dados pequena	0,83
RUSE – Reutilizabilidade Necessária	Médio	Aproveitamento em projetos	1,00
PDIF – Dificuldades com a Plataforma	Baixo	Não há restrições de tempo de execução, utilização de memória e a plataforma é considerada estável.	0,87
PERS –Capacidade do Pessoal	Muito Alto	Baixa rotatividade de pessoal e analistas e programadores com boa capacidade	0,63
PREX – Experiência Profissional do Pessoal	Alto	Experiência média de 2 anos com plataforma, aplicação e linguagem	0,87
FCIL – Facilidades	Médio	Uso de ferramentas CASE básicas e pouco apoio para desenvolvimento multilocal complexo.	1,00
SCED – Prazo necessário para o Desenvolvimento	Médio	Não há restrição de tempo de desenvolvimento	1,00

Tabela A2.11 – Valores dos direcionadores de custos para o exemplo apresentado

Para proceder no cálculo do esforço, utilizamos a fórmula a seguir:

$$E_{(P/M)} = A \times tamanho^{G} \times \sum_{i=1}^{i=n} ME_i$$

Onde:

- **A** = 2,94 (constante).
- O **tamanho** do *software* foi calculado em pontos de função no exemplo apresentado no Anexo 1. É importante observar que aqui deve ser considerado o número de pontos de função **não** ajustados, que para o exemplo em questão vale 23. O tamanho em pontos de função deve ser transformado em número de linhas de código fonte, utilizando-se tabelas de relação entre um ponto de função e o número de linhas de código fonte correspondente em uma determinada linguagem. Supondo que o sistema aqui exemplificado seja implementado em *Power Builder*, utilizando a tabela de conversão encontrada em BOEHM *et al.* (2000), tem-se que um ponto de função não ajustado equivale a 16 linhas de código fonte em *Power Builder*. Sendo assim, **tamanho** = 368 SLOC = 0,368 KLOC .
- $G = B + 0{,}01 \times \sum_{i=1}^{i=5} FE_i$ = 0,91 + 0,01 x 8,43 = 0,9943

Assim, obtemos:

$$E_{(P/M)} = A \times tamanho^{G} \times \sum_{i=1}^{i=n} ME_i$$

$$E_{(P/M)} = 2{,}94 \times 0{,}368^{0{,}9943} \times 5{,}2$$

$$E_{(P/M)} = \mathbf{5{,}66}$$

(iii) *Realizar Estimativas de Prazo*

Para calcular o prazo, utilizamos a seguinte fórmula:

$$P_{(M)} = C \times E^{F}$$

Onde:

- **C** = 3,67 (constante).
- $F = D + 0{,}2 \times (G - B)$ = 0,28 + 0,2 x (0,9943 – 0,91) = 0,3.
- **E** = 5,66

Assim:

$$P_{(M)} = C \times E^{F}$$

$$P_{(M)} = 3{,}67 \times 5{,}66^{\,0{,}3}$$

$$P_{(M)} = \mathbf{6{,}17}$$

Como não há restrições de tempo e SCED % = 100 não há necessidade de considerá-lo no cálculo do prazo, pois 100/100 = 1.

(iv) *Realizar Estimativas de Custos*

Inicialmente, é preciso obter o número médio de pessoas que serão alocadas à equipe do projeto. Para isso, basta dividir o valor do esforço pelo prazo. Assim,

equipe = 5,66 / 6,17 = 0,92 ≈ 1 pessoa.

Suponhamos que o salário médio dessa pessoa é R$1200,00

Para calcular os custos (C) basta multiplicar o salário do recurso pelo seu tempo de utilização, assim:

$C_{(R\$)}$ = 1200 x 5,66 = 6792

A2. 6 Considerações

Os valores encontrados com a utilização do COCOMO II foram relativamente altos para um sistema hipotético simples como o apresentado, o que demonstra que calibrar o modelo com dados próprios é necessário para torná-lo mais eficiente a uma organização específica. Se observarmos com atenção, é possível perceber que os valores das constantes representam alta influência nos resultados e, se estas constantes (bem como os valores dos direcionadores e fatores de equilíbrio) forem recalibradas, observando projetos da organização, as estimativas realizadas serão mais acuradas.

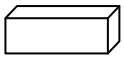
A descrição detalhada do processo de calibração do modelo para uma organização específica pode ser encontrado em BOEHM *et al.* (2000).

Anexo 3

Notação dos Diagramas de Workflow

Este anexo apresenta a notação utilizada nos diagramas de workflows.

Processo

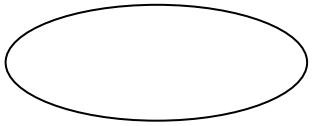


Evento

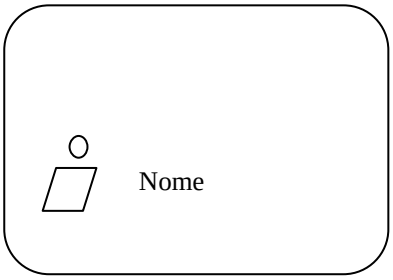
Evento dispara ou encerra um processo.

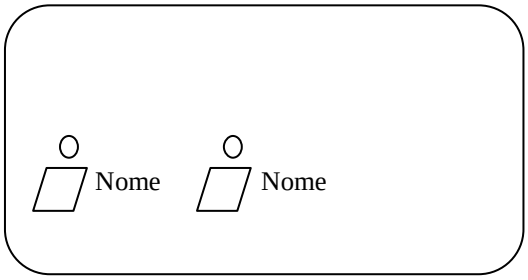
Grupo de Processos Relacionados

Este elemento serve para agrupar processos em categorias de processos.



Ator





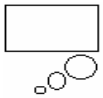
Atividade

Atividade Atômica

Atividade Composta

Atividade Externa

Conhecimento Explícito

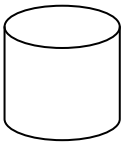


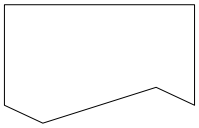
Conhecimento Tácito

Habilidade

Software ***Repositório*** ***Documento***

Nome



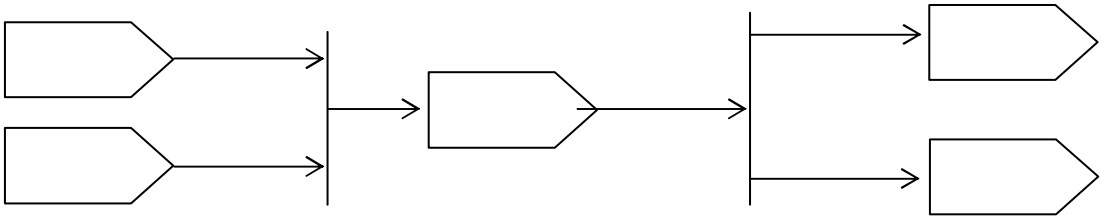


Operações Lógicas

AND

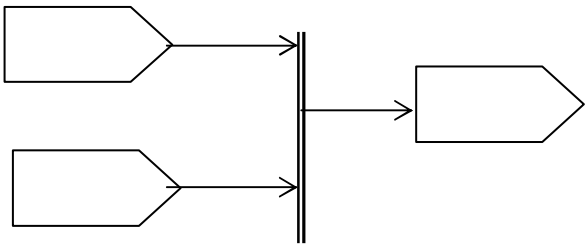
OR

XOR

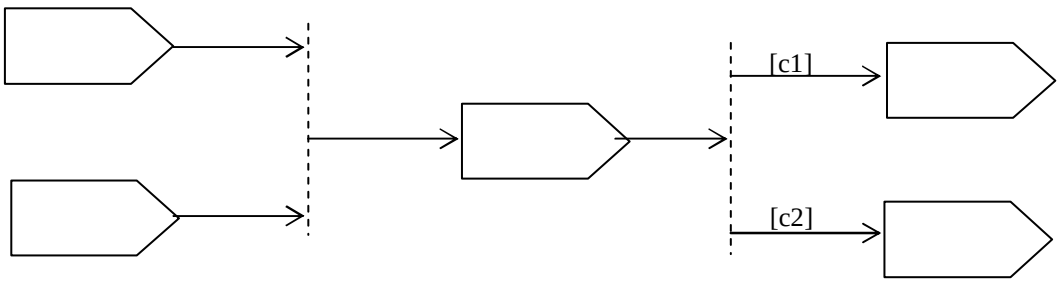


AND-Join AND-Split

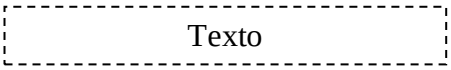
OR-Join



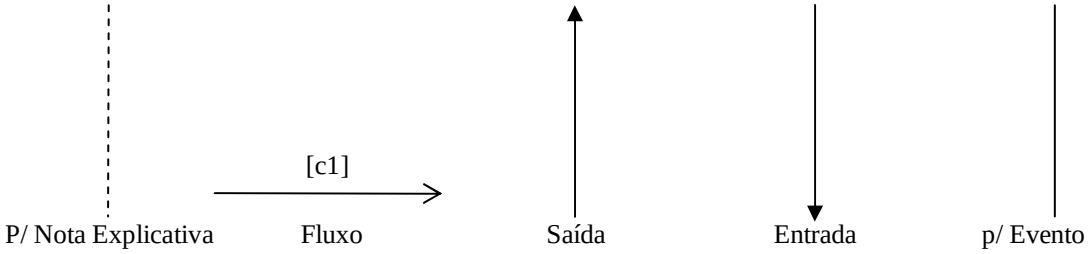
XOR-Join XOR-Split



Nota Explicativa



Associação



Anexo 4

Pesquisa de Dependências Usuais entre as Atividades do Projeto

Este anexo apresenta a pesquisa realizada para definir um conjunto de dependências usuais entre as atividades do processo de desenvolvimento.

A4. 1 Introdução

Como apresentado na definição do processo de gerência de tempo, no capítulo 4, uma das atividade do planejamento do cronograma de um projeto é determinar as relações de dependência entre as atividades do projeto. Na abordagem proposta por este trabalho, essa atividade será executada com o apoio de um conjunto de dependências usuais entre as atividades do projeto que poderão ser consultadas pelo gerente do projeto. Para definir esse conjunto de dependências, uma pesquisa foi realizada.

Na seção A4.2 é apresentado o planejamento da pesquisa e o formulário utilizado na coleta de dados; na seção A4.3 é apresentada a análise dos dados obtidos e na seção 4.4 os resultados da pesquisa.

A4. 2 Planejamento da Pesquisa

Descrição do Problema

O processo de gerência de tempo tem como principal objetivo elaborar e controlar o cronograma do projeto. O primeiro passo para a elaboração do cronograma é a determinação das dependências entre as atividades que compõem o processo de desenvolvimento do projeto. Dependências mal identificadas podem comprometer o desenvolvimento do projeto.

Para apoiar o gerente de projetos na determinação das relações de dependência entre as atividades do projeto, dados de dependências usuais são consideravelmente úteis (PMBOK, 2000). Este estudo visa caracterizar as relações de dependência usuais entre as atividades de um projeto, utilizando como base as atividades do processo de desenvolvimento da NBR ISO/IEC 12207.

Objetivo Global

Caracterizar as relações de dependência entre as atividades do processo de desenvolvimento da NBR ISO/IEC 12207.

Objetivo da Medição

Tendo como base uma proposta inicial das dependências entre as atividades do processo de desenvolvimento da NBR ISO/IEC 12207, caracterizar:

1. As relações de dependência entre as atividades.

1.1. Quais são as dependências que devem ser excluídas do conjunto de dependências inicial.

1.2. Quais são as dependências que devem ser incluídas no conjunto de dependências inicial.

É importante destacar que o conjunto inicial de dependências não estará disponível aos gerentes, procurando evitar, assim, que os mesmos sejam influenciados pelas dependências presentes no conjunto inicial.

Objetivo do estudo

Analisar as atividades do processo de desenvolvimento da NBR ISO/IEC 12207.

Com o propósito de caracterizar

Com respeito às relações de dependência entre as atividades

Do ponto de vista de gerentes de projeto

No contexto de gerência de tempo de projetos de *software*.

Objeto de estudo: Atividades do processo de desenvolvimento da NBR ISO/IEC 12207

Propósito: Caracterizar as relações de dependência entre as atividades pertencentes ao conjunto inicial de dependências.

Foco da qualidade: O foco da qualidade está na experiência do gerente de projetos em determinar as relações de dependência entre as atividades do processo de desenvolvimento da NBR ISO/IEC 12207.

Perspectiva: A perspectiva é do gerente de projetos de *software*

Contexto: O estudo será realizado utilizando gerentes de projeto de *software* com experiência em gerência de projetos de *software*. As atividades da NBR ISO/IEC 12207 serão apresentadas em um questionário a ser entregue aos participantes do estudo que indicarão as relações de dependências entre elas, segundo sua perspectiva.

Questões:

Q1: Existem dependências que devem ser incluídas no conjunto de dependências?
Métrica: A lista de dependências sugeridas pelos gerentes de projeto.
Q2: Existem dependências que devem ser excluídas do conjunto de dependências?
Métrica: A lista de dependências sugeridas pelos gerentes de projeto.

Definição das Hipóteses

Hipótese nula (H0): O conjunto de dependências inicial é completo, ou seja, não há dependências a serem retiradas nem incluídas.
Dp – conjunto de dependências inicial
Dr – dependências a serem retiradas do conjunto de dependências inicial
Di – dependências a serem incluídas no conjunto de dependências inicial
H0 : $Dr = Di = \varnothing$

Hipótese Alternativa (H1): Existem dependências a serem retiradas do conjunto de dependências inicial.
H1: $Dr \neq \varnothing$

Hipótese Alternativa (H2): Existem dependências a serem incluídas no conjunto de dependências inicial.
H2: $Di \neq \varnothing$

Seleção do Contexto

O estudo será conduzido de forma off-line, ou seja, o questionário será entregue aos participantes e não será acompanhado. Cada gerente de projeto terá o seu tempo e ambiente para preencher o questionário, colaborando com o estudo. Os participantes serão profissionais com experiência na área de gerência de projetos de *software*.

Seleção dos Indivíduos

Os indivíduos são selecionados baseados em conveniência e disponibilidade, isto é, serão selecionados os gerentes de projeto de *software* que participam de projetos da COPPETEC e/ou de outras empresas e que são conhecidos por sua experiência e atuação na área de gerência de projetos de *software*. Os indivíduos selecionados representam uma amostra dos gerentes de projeto de *software*, mas não são escolhidos de forma aleatória.

Validade

Validade Interna

Os participantes do estudo serão selecionados tendo como base sua experiência em gerência de projetos de *software*. Assume-se que eles são representativos para a população dos gerentes de projeto. Dessa forma, as relações de dependência indicadas pelos participantes serão baseadas na experiência pessoal de cada gerente de projeto.

Validade de conclusão

Hipótese Nula: A verificação da hipótese nula será feita por meio de simples demonstração de presença ou não de dependências na lista de dependências fornecida pelos gerentes.

A inclusão ou exclusão de dependências levará em consideração:

1 – O número de participantes que indicaram que uma dada dependência deveria ser incluída ou excluída;

2 – O nível de experiência do gerente de projeto que indicou que a dependência deveria ser incluída ou excluída.

Validade externa

Os participantes são representativos para a população dos gerentes de projeto de *softwar*e. A fim de avaliar o nível de experiência nas áreas de gerência de projetos e o conhecimento da norma NBR ISO/IEC 12207, os dados da caracterização de indivíduos podem ser analisados.

Instrumentação

O instrumento de pesquisa utilizado é apresentado a seguir.

Descrição da Instrumentação: Relações de Dependência entre as atividades do Processo de Desenvolvimento da NBR ISO/IEC 12207

O principal objetivo da gerência de tempo é planejar e controlar o cronograma de um projeto. O primeiro passo desse processo é a determinação das dependências entre as atividades do projeto.

Este questionário visa identificar as dependências usuais entre as atividades do processo de desenvolvimento de software, baseando-se no conhecimento de gerentes de projetos no que diz respeito às atividades do processo de desenvolvimento da NBR ISO/IEC 12207 - Tecnologia de Informação – Processos de ciclo de vida de software. O questionário faz parte de uma tese de mestrado da COPPE/UFRJ.

Caracterização do Participante:

Nome:					e-mail:		
ÁREA DE ATUAÇÃO				SOFTWARE QUE DESENVOLVE			
Empresa		*Universidade*			Administração		Gerenciador de Informações
	Empresário		Professor		Agropecuária		Gerenciador de Redes
	Gerente de Informática		Pesquisador		Automação		Saude
	Gerente da Qualidade		Consultor		Educacional		Software Gráfico
	Gerente de Projeto		Aluno de Doutorado		Entretenimento		Teleinformática/Telecomunicações
	Analista de Sistemas		Aluno de Mestrado		Ferramenta de Desenvolvimento de Sw		Transportes
	Outro:		Aluno de Graduação		Financeiro		Outros:
Tempo de Atuação na Área: ____ Anos Número de Projetos que já gerenciou: ____				*Desenvolve para:* ( ) Uso Próprio ( ) Clientes ( ) Pacote			
FORMAÇÃO				EXPERIÊNCIA EM PROCESSOS DE SOFTWARE			
Nível				Como você classificaria seu entendimento / experiência em Processos de Software?			
	Doutorado ( ) Eng de Software ( ) Computação/Informática ( ) Outro				Excelente		Nenhum
	Mestrado ( ) Eng de Software ( ) Computação/Informática ( ) Outro				Alto		
	Especialização ( ) Eng de Software ( ) Computação/Informática ( ) Outro				Médio		
	Graduação ( ) Eng de Software ( ) Computação/Informática ( ) Outro				Baixo		

CONHECIMENTO SOBRE A NBR ISO/IEC 12207	
Como você classificaria seu conhecimento sobre a NBR/ISO IEC 12207- **Tecnologia de Informação - Processos de ciclo de vida de software?**	
	Especialista
	Conhece
	Conhece, mas não usa
	Desconhece

Contato:
Ana Regina Rocha/Guilherme Horta Travassos / Monalessa Perini Barcellos
COPPE/UFRJ – Sistemas

Caixa Postal 68511
CEP 21945-970
e-mail: darocha@cos.ufrj.br / ght@cos.ufrj.br / mona@cos.ufrj.br

Muito obrigada por sua colaboração ao nosso questionário

INSTRUÇÕES
1 - Considere as atividades do processo de desenvolvimento da NBR ISO/IEC 12207 exibidas a seguir. Para cada atividade do processo, identifique com um X suas pré-atividades, ou seja, as atividades que precisam estar concluídas para que a atividade em análise seja executada.
Utilize a tabela "Comentários Adicionais" caso deseje registrar alguma observação.

Atividades do Processo de Desenvolvimento da NBR ISO/IEC 12207 -- Tecnologia de Informação - Processos de ciclo de vida de software que devem ser executadas	Atividades que precisam estar concluídas												
	Análise dos Requisitos do Sistema	Análise dos Requisitos do Software	Apoio à aceitação do Software	Codificação e Testes do Software	Implementação do processo	Instalação do Software	Integração do Sistema	Integração do Software	Projeto da Arquitetura do Sistema	Projeto da Arquitetura do Software	Projeto Detalhado do Software	Teste de Qualificação do Sistema	Teste de Qualificação do Software
Análise dos Requisitos do Sistema: descrição das funções, capacidades, requisitos e restrições do sistema.	■												
Análise dos Requisitos do Software: descrição das funções, capacidades, requisitos e restrições de cada item de software do sistema.		■											
Apoio à aceitação do Software: acompanhamento à utilização do software.			■										
Codificação e Testes do Software: implementação e testes do software e bases de dados.				■									
Implementação do processo : Definição do modelo de ciclo de vida, atividades e tarefas do projeto.					■								
Instalação do Software: implantação do software no ambiente do cliente.						■							
Integração do Sistema: integração dos itens de software ao sistema.							■						
Integração do Software: integração dos componentes que compõem cada item de software.								■					
Projeto da Arquitetura do Sistema: definição dos itens de hardware, software e operações do sistema.									■				
Projeto da Arquitetura do Software: definição dos itens de hardware, software e operações do sistema de cada item de software do sistema.										■			
Projeto Detalhado do Software: refinamento do projeto da arquitetura de cada componente de software, interface e bases de dados.											■		
Teste de Qualificação do Sistema: realização de testes segundo os requisitos de qualidade estabelecidos para o sistema.												■	
Teste de Qualificação do Software: realização de testes segundo os requisitos de qualidade estabelecidos para cada item de software.													■

Comentários Adicionais:

Conjunto inicial de Dependências

A seguir é apresentado o conjunto inicial de dependências proposto por este estudo.

Atividades do Processo de Desenvolvimento da NBR ISO/IEC 12207 -- Tecnologia de Informação – Processos de ciclo de vida de software que devem ser executadas	Atividades que precisam estar concluídas												
	Análise dos Requisitos do Sistema	Análise dos Requisitos do Software	Apoio à aceitação do Software	Codificação e Testes do Software	Implementação do processo	Instalação do Software	Integração do Sistema	Integração do Software	Projeto da Arquitetura do Sistema	Projeto da Arquitetura do Software	Projeto Detalhado do Software	Teste de Qualificação do Sistema	Teste de Qualificação do Software
Análise dos Requisitos do Sistema: descrição das funções, capacidades, requisitos e restrições do sistema.					X								
Análise dos Requisitos do Software: descrição das funções, capacidades, requisitos e restrições de cada item de software do sistema.	X				X								
Apoio à aceitação do Software: acompanhamento à utilização do software.	X	X		X	X	X	X	X	X	X	X	X	X
Codificação e Testes do Software: implementação e testes do software e bases de dados.	X	X			X				X	X	X		
Implementação do processo : Definição do modelo de ciclo de vida, atividades e tarefas do projeto.													
Instalação do Software: implantação do software no ambiente do cliente.	X	X		X	X		X	X	X	X	X	X	X
Integração do Sistema: integração dos itens de software ao sistema.	X	X		X	X			X	X	X	X		X
Integração do Software: integração dos componentes que compõem cada item de software.	X	X		X	X				X	X	X		
Projeto da Arquitetura do Sistema: definição dos itens de hardware, software e operações do sistema.	X				X								
Projeto da Arquitetura do Software: definição dos itens de hardware, software e operações do sistema de cada item de software do sistema.	X	X			x				X				
Projeto Detalhado do Software: refinamento do projeto da arquitetura de cada componente de software, interface e bases de dados.	X	X			X				X	x			
Teste de Qualificação do Sistema: realização de testes segundo os requisitos de qualidade estabelecidos para o sistema.	X	X		X	X		X	X	X	X	X		X
Teste de Qualificação do Software: realização de testes segundo os requisitos de qualidade estabelecidos para cada item de software.	X	X		X	X			X	X	X	X		

Tabela A4.1 – Conjunto inicial das dependências usuais entre as atividades do processo de desenvolvi mento

A4.3 Avaliação dos Dados Obtidos

Para coletar os dados necessários à definição das dependências usuais entre as atividades do processo de desenvolvimento de *software*, objetivo desta pesquisa, o formulário elaborado como forma de instrumentação foi entregue aos participantes e os dados obtidos foram organizados em uma planilha para serem analisados.

A análise dos dados das características dos indivíduos mostra uma heterogeneidade no que diz respeito à experiência em gerência de projetos de *software* e em processos de *software*. A tabela A4.2 apresenta a caracterização parcial dos participantes do estudo, exibindo o tempo de atuação na área, o número de projetos gerenciados, a experiência em processos de *software* e o conhecimento sobre a NBR ISO/IEC 12207 de cada indivíduo. A coluna Id descreve o identificador numérico atribuído a cada participante.

Id	Tempo de Atuação	Número de Projetos que já gerenciou	Experiência em Processos de Software	Conhecimento sobre a NBR ISO/IEC 12207
1	20	15	Excelente	Especialista
2	20	30	Excelente	Especialista
3	17	50	Excelente	Conhece
4	22	12	Excelente	Conhece
5	5	2	Médio	Conhece
6	8	6	Médio	Conhece
7	8	20	Alto	Conhece, mas não usa
8	3	4	Alto	Conhece, mas não usa

Tabela A4.2 – Caracterização Parcial dos Participantes do Estudo

Para diferenciar as respostas dos indivíduos, será atribuído um peso a cada um deles, considerando o *tempo de atuação na área*, o *número de projetos gerenciados*, a *experiência em processos de software* e o *conhecimento sobre a NBR ISO/IEC 12207*. A forma utilizada para distinguir a opinião dos gerentes foi baseada na proposta de FARIAS (2002), acrescentando-se o conhecimento sobre a norma NBR ISO/IEC 12207 do participante. A fórmula utilizada para definir o peso atribuído a um participante é

$$P(i) = \frac{TA(i)}{MedianaTA} + \frac{NP(i)}{MedianaNP} + f(i) \quad + g(i)$$

Onde:

- P(i) é o peso atribuído ao participante i;
- TA(i) é o tempo de atuação na área do participante i;
- MedianaTA é a mediana do tempo de atuação, considerando o tempo de atuação de cada participante do estudo;
- NP(i) é o número de projetos que o participante i já gerenciou;
- MedianaNP é a mediana do número de projetos gerenciados, considerando o número de projetos gerenciados por cada participante do estudo;
- f(i) = 0, se a experiência em processos de *software* do participante i for nenhuma ou baixa;
- f(i) = 1, se a experiência em processos de *software* do participante i for média;
- f(i) = 2, se a experiência em processos de *software* do participante i for alta;
- f(i)= 3, se a experiência em processos de *software* do participante i for excelente.
- g(i) = 0, se a classificação do conhecimento do participante i no que diz respeito à norma NBR ISO/IEC 12207 for 'desconhece';
- g(i) = 1, se a classificação do conhecimento do participante i no que diz respeito à norma NBR ISO/IEC 12207 for 'conhece, mas não usa';
- g(i) = 2, se a classificação do conhecimento do participante i no que diz respeito à norma NBR ISO/IEC 12207 for 'conhece';
- g(i)= 3, se a classificação do conhecimento do participante i no que diz respeito à norma NBR ISO/IEC 12207 for 'especialista';

Analisando os dados da caracterização dos indivíduos (tabela A4.1), pode-se constatar que a mediana do tempo de atuação é igual a 12,5 e a mediana do número de projetos gerenciados igual a 13,5. Tomando por base esses valores e os dados da tabela A4.2, o peso de cada participante é calculado considerando a fórmula descrita anteriormente. A tabela A4.3 apresenta o peso atribuído a cada indivíduo participante da pesquisa.

Id	Peso
1	8,71
2	9,82
3	10,06
4	7,65
5	3,55
6	4,08
7	5,12
8	3,54

Tabela A4.3 – Pesos dos participantes do estudo

A figura A4.1 exibe o gráfico que descreve, no eixo horizontal, os participantes do estudo, e, no eixo vertical, o peso de cada participante.

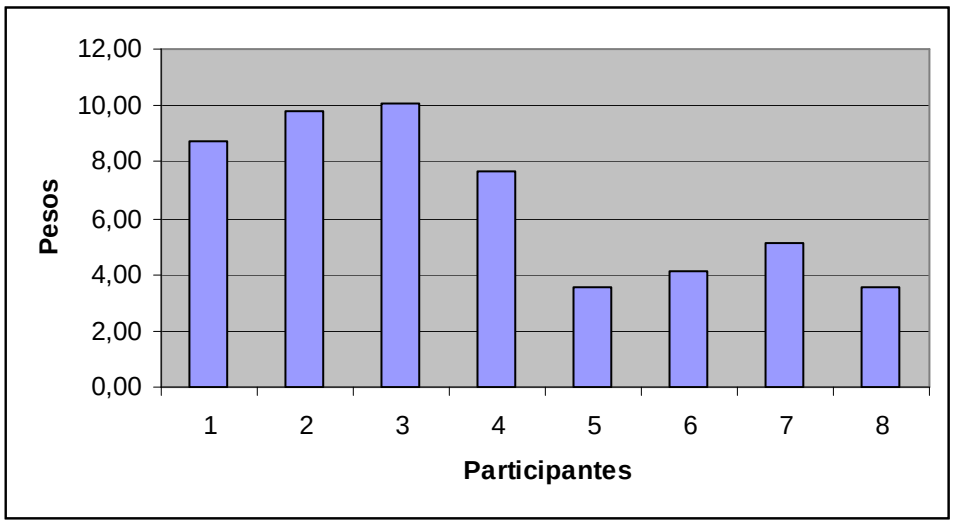


Figura A4.1 – Gráfico Participantes x Pesos

Analisando o gráfico é possível perceber dois grandes grupos de participantes: os participantes identificados por 1, 2, 3 e 4 são considerados *experientes* e têm seus pesos na faixa de 7 a 11. Os participantes 5, 6, 7 e 8 são *pouco experientes* e têm seus pesos na faixa de 3 a 6. Não foram identificados indivíduos *inexperientes*, que possuiriam seu peso inferior a 3.

Para definir o conjunto de dependências usuais entre as atividades do processo de desenvolvimento de *software*, as dependências pertencentes ao conjunto inicial proposto serão verificadas considerando-se o número de votos obtidos e o peso atribuído a cada voto. Para concluir se uma dependência deve ou não pertencer ao

conjunto resultante desta pesquisa é necessário estabelecer um ponto de inclusão, ou seja, o valor a partir do qual pode-se considerar um elemento como pertencente ao resultado final.

Se todos os participantes do estudo identificarem uma determinada dependência, teremos o valor de votos igual a 52,54[13] para essa dependência. Esse valor representa o valor máximo de votos para uma determinada dependência da pesquisa. Assim, o ponto de inclusão é estabelecido em 26,27, que representa 50% do valor máximo de voto para uma determinada dependência.

Após a definição do ponto de inclusão, a análise e interpretação dos dados obtidos é concluída e o conjunto de dependências usuais final é estabelecido levando-se em consideração o peso atribuído a cada participante da pesquisa.

A seguir são apresentados os votos para as dependências de cada atividade do processo de desenvolvimento de *software* da NBR ISO/IEC 12207, utilizado como base para esta pesquisa.

Para melhor visualizar os resultados, representaremos as atividades pela seguinte identificação:

Identificação	**Atividade**
1	Análise de Requisitos de Sistema
2	Análise de Requisitos do *Software*
3	Apoio à Aceitação do *Software*
4	Codificação e Testes do *Software*
5	Implementação do Processo
6	Instalação do *Software*
7	Integração do Sistema
8	Integração do *Software*
9	Projeto da Arquitetura do Sistema
10	Projeto da Arquitetura do *Software*
11	Projeto Detalhado do *Software*
12	Teste de Qualificação do Sistema
13	Teste de Qualificação do *Software*

Tabela A4.4 – Identificação das Atividades do Processo de Desenvolvimento

Para cada atividade, as colunas das dependências que estão presentes no conjunto inicial proposto serão representadas em cinza. Assim, ficará mais claro observar as diferenças entre as dependências propostas pelo conjunto inicial e as identificadas pelos participantes. È importante lembrar que os participantes não tiveram acesso ao conjunto inicial proposto.

A tabela A4.5 apresenta os dados obtidos para as dependências da atividade *Análise de Requisitos do Sistema*.

Id	**Dependências**												
Participante	1	2	3	4	5	6	7	8	9	10	11	12	13
1					8,71								
2					9,82								
3					10,06								
4									7,65				
5					3,55								
6													
7													
8													

Tabela A4.5 – Dados obtidos para as dependências da atividade Análise de Requisitos do Sistema

Analisando os dados apresentados na tabela, podemos observar que a dependência 5 está presente no conjunto de dependências usuais, pois recebeu número de votos igual a 32,14, que é superior ao ponto de inclusão.

A dependência 9, apesar de ter sido identificada por um dos participantes, não será incluída no conjunto de dependências usuais, uma vez que não alcançou o número de votos suficiente.

A tabela A4.6 apresenta os dados obtidos para as dependências da atividade *Análise de Requisitos do Software*.

[13] Somatório dos pesos dos participantes: 8,71 + 9,82 + 10,06 + 7,65 + 3,55 + 4,08 + 5,12 + 3,54 = 52,54

Id Participante	Dependências												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71				8,71								
2	9,82				9,82				9,82				
3	10,06				10,06								
4	7,65				7,65				7,65				
5	3,55				3,55								
6	4,08												
7													
8	3,54												

Tabela A4.6 – Dados obtidos para as dependências da atividade Análise de Requisitos do Software

Analisando os dados apresentados na tabela A4.6, observamos que as dependências 1 e 5 serão incluídas no conjunto final de dependências usuais pois alcançaram valor de votos superior ao valor do ponto de inclusão.

Já a dependência 9, que foi identificada por dois participantes, não será incluída no conjunto final de dependências usuais, pois alcançou 17,47 votos, o que não é suficiente para ser considerada elemento do conjunto final.

A tabela A4.7 apresenta os dados obtidos para as dependências da atividade *Apoio à Aceitação do Software*

Id Participante	Dependências												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71		8,71	8,71	8,71	8,71	8,71	8,71	8,71	8,71	8,71	8,71
2	9,82	9,82		9,82	9,82	9,82	9,82	9,82	9,82	9,82	9,82	9,82	9,82
3	10,06	10,06		10,06	10,06	10,06	10,06	10,06	10,06	10,06	10,06	10,06	10,06
4	7,65	7,65		7,65	7,65	7,65	7,65	7,65	7,65	7,65	7,65	7,65	7,65
5	3,55	3,55		3,55	3,55	3,55	3,55	3,55	3,55	3,55	3,55	3,55	3,55
6	4,08	4,08		4,08	4,08	4,08	4,08	4,08	4,08	4,08	4,08	4,08	4,08
7	5,12	5,12			5,12								
8	3,54	3,54		3,54	3,54	3,54		3,54	3,54	3,54	3,54		3,54

Tabela A4.7 – Dados obtidos para as dependências da atividade Apoio à Aceitação do Software

Analisando a tabela A4.7, podemos perceber que as dependências 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12 e 13 serão incluídas no conjunto final de dependências, pois obtiveram valor de votos superior ao valor do ponto de inclusão.

A tabela A4.8 apresenta os dados obtidos para as dependências da atividade *Codificação e Testes do Software*

Id Participante	**Dependências**												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71			8,71				8,71	8,71	8,71		
2	9,82	9,82			9,82				9,82	9,82	9,82		
3	10,06	10,06			10,06				10,06	10,06	10,06		
4	7,65	7,65			7,65	7,65		7,65	7,65	7,65	7,65		
5	3,55	3,55			3,55				3,55	3,55	3,55		
6	4,08	4,08			4,08				4,08	4,08	4,08		
7	5,12	5,12							5,12	5,12	5,12		
8	3,54	3,54			3,54					3,54	3,54		

Tabela A4.8 – Dados obtidos para as dependências da atividade Codificação e Testes do Software

Observando os dados apresentados na tabela A4.8, notamos que as dependências 1, 2, 5, 9, 10 e 11 serão incluídas no conjunto final de dependências, uma vez que obtiveram valor de votos superior ao valor do ponto de inclusão.

As dependências 6 e 8, apesar de terem sido identificadas, não pertencerão ao conjunto final pois não alcançaram votos suficientes para tal.

A tabela A4.9 apresenta os dados obtidos para as dependências da atividade *Implementação do Processo*

Id Participante	Dependências												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1													
2													
3													
4	7,65												
5													
6	4,08	4,08											
7	5,12	5,12											
8		3,54											

Tabela A4.9 – Dados obtidos para as dependências da atividade Implementação do Processo

Para a atividade *Implementação do Processo* nenhuma dependência será incluída no conjunto final de dependências, pois as dependências identificadas pelos participantes para essa atividade não alcançaram votos suficientes para pertencer ao conjunto final.

A tabela A4.10 apresenta os dados obtidos para as dependências da atividade *Instalação do Software*

Id Participante	Dependências												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71		8,71	8,71		8,71	8,71	8,71	8,71	8,71	8,71	8,71
2	9,82	9,82		9,82	9,82		9,82	9,82	9,82	9,82	9,82	9,82	9,82
3	10,06	10,06		10,06	10,06		10,06	10,06	10,06	10,06	10,06	10,06	10,06
4	7,65	7,65	7,65	7,65	7,65		7,65	7,65	7,65	7,65	7,65	7,65	7,65
5	3,55	3,55		3,55	3,55		3,55	3,55	3,55	3,55	3,55	3,55	3,55
6	4,08	4,08		4,08	4,08		4,08	4,08	4,08	4,08	4,08		4,08
7	5,12	5,12	5,12	5,12	5,12	5,12	5,12	5,12	5,12	5,12	5,12	5,12	5,12
8	3,54	3,54	3,54	3,54	3,54			3,54	3,54	3,54	3,54		3,54

Tabela A4.10 – Dados obtidos para as dependências da atividade Instalação do Software

Analisando a tabela A4.10, podemos observar que as dependências 1, 2, 4, 5, 7, 8, 9, 10, 11, 12, e 13 serão incluídas no conjunto final de dependências, uma vez que obtiveram o valor máximo de votos dos participantes.

As dependências 3 e 6, apesar de terem sido identificadas, não pertencerão ao conjunto final pois não alcançaram votos suficientes para tal.

A tabela A4.11 apresenta os dados obtidos para as dependências da atividade *Integração do Sistema*

Id Participante	**Dependências**												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71		8,71	8,71			8,71	8,71	8,71	8,71		
2	9,82	9,82		9,82	9,82			9,82	9,82	9,82	9,82		9,82
3	10,06	10,06		10,06	10,06			10,06	10,06	10,06	10,06		10,06
4	7,65	7,65		7,65	7,65			7,65	7,65	7,65	7,65		7,65
5	3,55	3,55		3,55	3,55			3,55	3,55	3,55	3,55		3,55
6	4,08	4,08		4,08	4,08			4,08	4,08	4,08	4,08		
7	5,12	5,12		5,12	5,12			5,12	5,12	5,12	5,12		
8	3,54	3,54		3,54	3,54			3,54	3,54	3,54	3,54		

Tabela A4.11 – Dados obtidos para as dependências da atividade Integração do Sistema

Observando os dados apresentados na tabela A4.11, percebemos que as dependências 1, 2, 4, 5, 8, 9, 10 e 11 serão incluídas no conjunto final de dependências, pois obtiveram o valor máximo de votos dos participantes.

A dependência 13 alcançou 31,08 votos, logo, também será incluída no conjunto final.

A tabela A4.12 apresenta os dados obtidos para as dependências da atividade *Integração do Software*

Id Participante	**Dependências**												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71		8,71	8,71				8,71	8,71	8,71		
2	9,82	9,82		9,82	9,82				9,82	9,82	9,82		
3	10,06	10,06		10,06	10,06				10,06	10,06	10,06		
4	7,65	7,65		7,65	7,65				7,65	7,65	7,65		
5	3,55	3,55		3,55	3,55				3,55	3,55	3,55		
6	4,08	4,08		4,08	4,08				4,08	4,08	4,08		
7	5,12	5,12		5,12	5,12				5,12	5,12	5,12		
8	3,54	3,54		3,54	3,54				3,54	3,54	3,54		

Tabela A4.12 – Dados obtidos para as dependências da atividade Integração do Software

Observando a tabela A4.12, notamos que as dependências 1, 2, 4, 5, 8, 9, 10 e 11 serão incluídas no conjunto final de dependências, uma vez que obtiveram o valor máximo de votos dos participantes.

A tabela A4.13 apresenta os dados obtidos para as dependências da atividade *Projeto da Arquitetura do Sistema*

Id Participante	**Dependências**												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71				8,71								
2	9,82				9,82								
3	10,06				10,06								
4	7,65	7,65			7,65								
5	3,55				3,55								
6	4,08	4,08			4,08								
7	5,12				5,12								
8	3,54	3,54											

Tabela A4.13 – Dados obtidos para as dependências da atividade Projeto da Arquitetura do Sistema

Observando os dados apresentados na tabela A4.13, podemos perceber que as dependências 1 e 5 serão incluídas no conjunto final de dependências, uma vez que obtiveram valor de votos superior ao valor do ponto de inclusão.

A dependência 2, apesar de ter sido identificada, não pertencerá ao conjunto final pois não alcançou votos suficientes para tal.

A tabela A4.14 apresenta os dados obtidos para as dependências da atividade *Projeto da Arquitetura do Software*

Id Participante	Dependências												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71			8,71				8,71				
2	9,82	9,82			9,82				9,82				
3	10,06	10,06			10,06				10,06				
4	7,65	7,65			7,65				7,65				
5	3,55	3,55			3,55				3,55				
6	4,08	4,08			4,08				4,08				
7	5,12	5,12											
8	3,54	3,54											

Tabela A4.14 – Dados obtidos para as dependências da atividade Projeto da Arquitetura do Software

Analisando a tabela A4.14, percebemos que as dependências 1, 2, 5 e 9 serão incluídas no conjunto final de dependências, uma vez que obtiveram valor de votos superior ao valor do ponto de inclusão.

A tabela A4.15 apresenta os dados obtidos para as dependências da atividade *Projeto da Detalhado do Software*

Id Participante	Dependências												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71			8,71				8,71	8,71			
2	9,82	9,82			9,82				9,82	9,82			
3	10,06	10,06			10,06				10,06	10,06			
4	7,65	7,65			7,65				7,65	7,65			
5	3,55	3,55			3,55				3,55	3,55			
6	4,08	4,08			4,08				4,08	4,08			
7	5,12	5,12											
8	3,54	3,54								3,54			

Tabela A4.15 – Dados obtidos para as dependências da atividade Projeto da Detalhado do Software

Observando os dados apresentados na tabela A4.15, nota-se que as dependências 1, 2, 5, 9 e 10 serão incluídas no conjunto final de dependências, pois obtiveram valor de votos superior ao valor do ponto de inclusão.

A tabela A4.16 apresenta os dados obtidos para as dependências da atividade *Teste de Qualificação do Sistema*

Id Participante	**Dependências**												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71		8,71	8,71		8,71	8,71	8,71	8,71	8,71		8,71
2	9,82	9,82		9,82	9,82		9,82	9,82	9,82	9,82	9,82		9,82
3	10,06	10,06		10,06	10,06		10,06	10,06	10,06	10,06	10,06		10,06
4	7,65	7,65	7,65	7,65	7,65	7,65		7,65	7,65	7,65	7,65		7,65
5	3,55	3,55		3,55	3,55		3,55	3,55	3,55	3,55	3,55		3,55
6	4,08	4,08		4,08	4,08		4,08	4,08	4,08	4,08	4,08		4,08
7	5,12	5,12		5,12	5,12		5,12	5,12	5,12	5,12	5,12		
8	3,54	3,54		3,54	3,54		3,54	3,54	3,54	3,54	3,54		3,54

Tabela A4.16 – Dados obtidos para as dependências da atividade Teste de Qualificação do Sistema

Observando os dados da tabela A4.16, notamos que as dependências 1, 2, 4, 5, 7, 8, 9, 10, 11 e 13 serão incluídas no conjunto final de dependências, uma vez que obtiveram valor de votos superior ao valor do ponto de inclusão.

As dependências 3 e 6, apesar de terem sido identificadas, não pertencerão ao conjunto final pois não alcançaram votos suficientes.

A tabela A4.17 apresenta os dados obtidos para as dependências da atividade *Teste de Qualificação do Software*

Id Participante	**Dependências**												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8,71	8,71		8,71	8,71		8,71	8,71	8,71	8,71	8,71		
2	9,82	9,82		9,82	9,82			9,82	9,82	9,82	9,82		
3	10,06	10,06		10,06	10,06		10,06	10,06	10,06	10,06	10,06		
4	7,65	7,65	7,65	7,65	7,65	7,65		7,65	7,65	7,65	7,65		
5	3,55	3,55		3,55	3,55			3,55	3,55	3,55	3,55		
6	4,08	4,08		4,08	4,08		4,08	4,08	4,08	4,08	4,08		
7	5,12	5,12		5,12	5,12			5,12	5,12	5,12	5,12		5,12
8	3,54	3,54		3,54	3,54			3,54	3,54	3,54	3,54		

Tabela A4.17 – Dados obtidos para as dependências da atividade Teste de Qualificação do Software

Analisando a tabela A4.17, percebemos que as dependências 1, 2, 4, 5, 8, 9, 10 e 11 serão incluídas no conjunto final de dependências, uma vez que obtiveram o valor máximo de votos dos participantes.

As dependências 3, 6, 7 e 13 apesar de terem sido identificadas, não pertencerão ao conjunto final pois não alcançaram votos suficientes para tal.

É importante notar que a utilização dos pesos dos participantes, calculado considerando o *tempo de atuação na área*, o *número de projetos gerenciados,* a *experiência em processos de software* e o *conhecimento sobre a NBR ISO/IEC 12207*, determinou a inclusão ou não de algumas dependências. Na atividade *Análise de Requisitos do Sistema* (tabela A4.5), quatro participantes identificaram a dependência 5. O mesmo ocorreu para a dependência 13 da atividade *Integração do Sistema* (tabela A4.11). Se tivesse sido utilizado o critério da maioria (metade dos participantes mais um) para a inclusão de uma dependência, as dependências 5 e 13, nos casos acima citados, não seriam incluídas no conjunto final de dependências. Porém, a consideração dos pesos dos participantes fez com que elas fossem incluídas no conjunto final, pois a soma dos pesos dos quatro participantes que identificaram as dependências superou o valor do ponto de inclusão.

A4.4 Resultados Obtidos

A pesquisa realizada atingiu os objetivos iniciais, uma vez que o conjunto de dependência usuais das atividades do processo de desenvolvimento foi caracterizado.

O conjunto de dependências final foi verificado avaliando-se as dependências identificadas pelos participantes que deveriam ser incluídas ou excluídas do conjunto inicial de dependências. A tabela A4.18 apresenta o conjunto de dependências resultante da análise dos dados obtidos.

Analisando a tabela A4.18 é possível perceber que o conjunto final de dependências é igual ao conjunto inicialmente proposto, ou seja, nenhuma dependência foi incluída ou excluída segundo o critério utilizado. Porém, a hipótese nula (H0) foi verificada como falsa, uma vez que existiram dependências sugeridas para inclusão que não estavam presentes no conjunto inicial, bem como existiram dependências sugeridas para exclusão do conjunto inicial de dependências.

Atividades do Processo de Desenvolvimento da NBR ISO/IEC 12207 -- Tecnologia de Informação – Processos de ciclo de vida de software que devem ser executadas	Atividades que precisam estar concluídas												
	Análise dos Requisitos do Sistema	Análise dos Requisitos do Software	Apoio à aceitação do Software	Codificação e Testes do Software	Implementação do processo	Instalação do Software	Integração do Sistema	Integração do Software	Projeto da Arquitetura do Sistema	Projeto da Arquitetura do Software	Projeto Detalhado do Software	Teste de Qualificação do Sistema	Teste de Qualificação do Software
Análise dos Requisitos do Sistema: descrição das funções, capacidades, requisitos e restrições do sistema.					X								
Análise dos Requisitos do Software: descrição das funções, capacidades, requisitos e restrições de cada item de software do sistema.	X				X								
Apoio à aceitação do Software: acompanhamento à utilização do software.	X	X		X	X	X	X	X	X	X	X	X	X
Codificação e Testes do Software: implementação e testes do software e bases de dados.	X	X			X				X	X	X		
Implementação do processo : Definição do modelo de ciclo de vida, atividades e tarefas do projeto.													
Instalação do Software: implantação do software no ambiente do cliente.	X	X		X	X		X	X	X	X	X	X	X
Integração do Sistema: integração dos itens de software ao sistema.	X	X		X	X			X	X	X	X		X
Integração do Software: integração dos componentes que compõem cada item de software.	X	X		X	X				X	X	X		
Projeto da Arquitetura do Sistema: definição dos itens de hardware, software e operações do sistema.	X				X								
Projeto da Arquitetura do Software: definição dos itens de hardware, software e operações do sistema de cada item de software do sistema.	X	X			x				X				
Projeto Detalhado do Software: refinamento do projeto da arquitetura de cada componente de software, interface e bases de dados.	X	X			X				X	x			
Teste de Qualificação do Sistema: realização de testes segundo os requisitos de qualidade estabelecidos para o sistema.	X	X		X	X		X	X	X	X	X		X
Teste de Qualificação do Software: realização de testes segundo os requisitos de qualidade estabelecidos para cada item de software.	X	X		X	X			X	X	X	X		

Tabela 4.18 – Conjunto final de dependências usuais entre as atividades do processo de desenvolvimento.

Ao realizar uma análise das dependências que foram sugeridas pelos participantes e que não foram incluídas no conjunto final pois não alcançaram o valor de votos suficiente, é possível perceber que, tanto os participantes do grupo *experiente* quanto os participantes do grupo *pouco experiente* identificaram dependências que não estavam presentes no conjunto inicial. Porém, ao analisar essas dependências, nota-se que as mesmas não são verdadeiras para um processo de desenvolvimento de *software*. Outra observação interessante é que dos 14 votos do grupo *experiente* registrados para dependências que não foram incluídas, 11 foram realizados pelo participante de menor peso do grupo. A tabela 4.19 apresenta o número de dependências que foram sugeridas, mas não incluídas, e o número de votos de cada grupo.

Nº de dependências sugeridas para inclusão que não alcançaram valor de votos suficiente	Nº de votos do grupo *experiente*	Nº de votos do grupo *pouco experiente*
13	14	13

Tabela A4.19 – Relação das dependências para inclusão com o nº de votos recebidos de cada grupo

Por outro lado, analisando-se as dependências que foram sugeridas para exclusão do conjunto inicial, ou seja, que não foram identificadas pelos participantes, é possível perceber que a maioria delas recebeu votos para exclusão dos participantes do grupo *pouco experiente*, com raras exceções. A tabela 4.20 apresenta o número de dependências que foram sugeridas, mas não incluídas, e o número de votos de cada grupo.

Nº de dependências sugeridas para exclusão que não alcançaram valor de votos suficiente	Nº de votos do grupo *experiente*	Nº de votos do grupo *pouco experiente*
24	2	37

Tabela A4.20 – Relação das dependências para exclusão como nº de votos recebidos de cada grupo

Sendo assim, podemos concluir que os participantes mais experientes registraram poucas alterações para o conjunto inicial de dependências e que, a maior parte das alterações foi sugerida pelos participantes menos experientes, sendo estas irreais quando analisadas no contexto do processo de desenvolvimento de *software*.

As características de alguns dos participantes podem ser consideradas para que seja feita uma análise do comportamento similar de seus resultados. Por exemplo, os três participantes do grupo *experiente* que apresentam maior peso possuem características que direcionam seus resultados a serem coincidentes, pois são profissionais que têm trabalhado seguindo uma mesma abordagem para processos de *software*. O participante de menor peso desse grupo, cujos resultados diferenciam-se dos demais, não tem participação ativa nos trabalhos realizados pelos demais elementos do grupo nesse contexto, o que pode justificar a diferença de seus votos em relação aos demais elementos do grupo.

No grupo *pouco experiente*, observa-se que os participantes de identificação 7 e 8 são os que sugerem o maior número de alterações. Analisando a tabela de caracterização parcial dos participantes (tabela A4.2) notamos que estes são os indivíduos com menor conhecimento sobre a norma NBR ISO/IEC 12207, o que pode justificar o número de alterações por eles sugeridas.

É importante ressaltar que esta pesquisa deve ser repetida, com um número maior de participantes e com perfis mais variados. No formulário que foi entregue aos participantes, um espaço foi destinado a comentários livres. Algumas considerações foram registradas e devem ser consideradas para a repetição da pesquisa, podendo até mesmo, ser necessário realizar algumas alterações para obter melhores resultados, como por exemplo, considerar iterações do processo e atividades que podem ser executadas em paralelo. A tabela A4.21 apresenta os comentários apresentados pelos participantes.

Id Participante	Comentários
1	“As atividades relacionadas a sistema são opcionais e podem não existir caso não exista especificação de requisitos do sistema.”
2	“Na prática, e a própria norma prevê, essas atividades se sobrepõem e são executadas de forma iterativa.”
3	“Dependendo da complexidade e outras características do projeto de software, a implementação do processo pode ser refinada após as fases “Análise dos Requisitos do Sistema” e “Análise dos Requisitos do Software”. Além disso, dependendo da metodologia e estratégia de desenvolvimento de software, os testes, a instalação do software e a sua aceitação pode ser realizada diversas vezes ao longo da implementação/integração de itens de software até culminar com a integração do sistema.”

Tabela A4.21 – Comentários registrados pelos participantes

A4.5 Lições Aprendidas

Visando melhorar a realização do estudo das dependências entre as atividades do processo de desenvolvimento de *software*, alguns pontos precisam ser revistos, já que foram considerados como limitações de sua primeira execução. Os pontos abaixo foram observados a partir de comentários, sugestões e/ou dúvidas dos participantes da pesquisa e podem ser tratados em uma repetição futura da pesquisa.

1. **Considerar atividades de outros modelos de processo**: Seria interessante realizar a pesquisa considerando as dependências entre as atividades do processo de desenvolvimento sugerido por outros modelos e normas (CMM e SPICE – ISO 15504 por exemplo). Assim, também seria possível realizar um estudo comparativo entre os resultados e propor um conjunto genérico de dependências usuais, uma vez que esta pesquisa propõe um conjunto de dependências baseado apenas nas atividades da NBR ISO/IEC 12207.

2. **Alterar a instrumentação**: Alguns participantes sentiram dificuldades em preencher o formulário, uma vez que este não considera as iterações de processo e realização de atividades em paralelo. Sendo assim, seria interessante alterar o formulário incluindo uma forma de registrar as situações acima mencionadas.

3. **Acompanhar o preenchimento do formulário:** Segundo o planejamento realizado, o estudo seria conduzido de forma *off-line*, ou seja, o questionário seria entregue aos participantes e o preenchimento do mesmo não seria acompanhado. Porém, dada a proximidade de um dos participantes, este foi acompanhado durante o preenchimento do questionário. A realização do acompanhamento mostrou que as contribuições para a pesquisa são maiores, uma vez que eventuais comentários ou sugestões podem ser registrados durante o preenchimento do formulário, enriquecendo os resultados. Além disso, durante o acompanhamento, é possível sanar dúvidas do participante que poderiam comprometer a qualidade de suas respostas.

A4.6 Considerações Finais

Este anexo apresentou o planejamento, análise dos resultados obtidos e o resultado da pesquisa de dependências usuais entre as atividades do processo de desenvolvimento de projetos de *software*. O objetivo da pesquisa foi caracterizar um conjunto de dependências usuais entre as atividades do processo de desenvolvimento de projetos de *software*, baseando-se nas atividades do processo de desenvolvimento da NBR ISO/IEC 12207.

Apesar do número reduzido de participantes, a pesquisa atingiu seu objetivo inicial considerando-se que os participantes possuem experiência em gerência de projetos e processos de *software*. O conjunto de dependências usuais resultante desta pesquisa estará armazenado no repositório da organização para que seja utilizado pelos gerentes de projeto durante o planejamento do cronograma do projeto.

Anexo 5

Modelo de Classes

Este anexo apresenta o modelo de classes da Estação TABA com as extensões incorporadas pelo modelo proposto nesta tese. São apresentadas as classes acrescidas para atender à ferramenta CustPlan e suas descrições.

A5.1 Caraterização de Projetos

A tabela A5.1 apresenta as classes que foram incluídas no modelo TABA para realizar a caracterização de projetos.

Classe	Descrição
CriterioCaracterizacao	*Critérios que caracterizam projetos de software na Estação TABA.*
ValorDominio	*Conjunto de valores que os critérios podem assumir.*
GrupoAplicacao	*Grupos de aplicação aos quais os critérios de caracterização podem se aplicar. Exemplos são equipe, cliente, gerente.*
PerfilProjeto	*Valor de um critério específico considerando um grupo de aplicação específico em um projeto específico.*
ConhecimentoIndustria	*Indústrias pertencente à classificação padrão de indústrias. Um projeto de software é destinado a uma indústria específica.*
ConhecimentoNatureza	*Todas as naturezas possíveis de um projeto de software.*
ConhecimentoEscopo	*Todas os possíveis escopos de um projeto de software.*

Tabela A5.1 – Classes para a Caracterização de Projetos

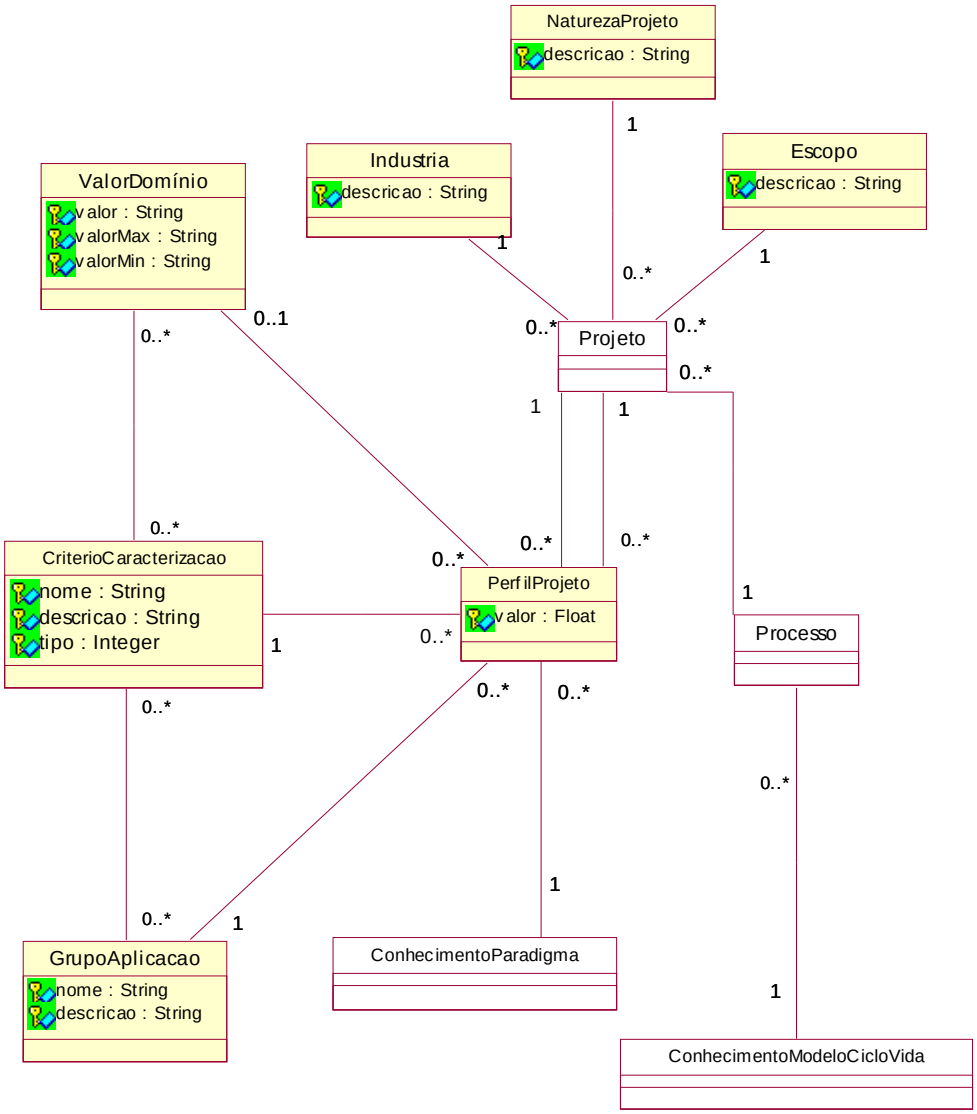


Figura A5.1 – Modelo para Caracterização de Projetos

A5.2 Planejamento de Tempo e Custos

A tabela A5.2 apresenta as classes que foram incluídas no modelo TABA para realizar o planejamento de tempo e custos.

Classe	*Descrição*
CaracteristicaSistema	*Armazena as 14 características de sistema a serem avaliadas na técnica Análise de Pontos de Função.*
NívelInfluencia	*Armazena os níveis de influência de cada característica do sistema.*
TipoFuncao	*Armazena as funções da Análise de Pontos de Função.*
Elemento	*Armazena os elementos da Análise de Pontos de Função.*
ElementoTipoFuncao	*Armazena a associação entre os elementos e as funções da Análise de Pontos de Função.*
ComplexidadeFuncional	*Armazena a relação das contagens dos elementos das funções com a complexidade e comtribuição para a Análise de Pontos de Função.*
InfluênciaCaracteristicaSistema	*Armazena a influência das características de sistema para o projeto.*
ValorPontoFuncao	*Armazena os valores dos parametros da Análise de Pontos de Função para o projeto.*
ModeloCOCOMOII	*Armazena os tipo de modelos do COCOMO II.*
ConstanteCOCOMOII	*Armazena as constantes utilizadas pelo COCOMO II.*
FatorEquilibrio	*Armazena os fatores de equilíbrio do COCOMO II.*
DirecionadorCusto	*Armazena os direcionadores de custos do COCOMO II.*
Escala	*Armazena a escala utilizada pelo COCOMO II.*
ItemEscala	*Armazena os itens da escala utilizados pelo COCOMO II.*
ValorCOCOMO II	*Armazena os valores do COCOMO II para o projeto.*
Modulo	*Armazena os módulos do projeto.*
Estimativa	*Armazena as estimativas realizadas para o projeto/módulo/atividade.*

Tabela A5.2 – Classes para o Planejamento de Custos

Classe	Descrição
ParametroEstimativa	*Armazena os parâmetros de estimativa disponíveis no TABA. Exemplos são: custos, esforço e prazo.*
FormaEstimativa	*Armazena as formas possíveis de estimativas, podendo ser ad hoc, baseada em analogias ou por métodos.*
GrupoEstimativa	*Armazena os grupos de estimativas do cronograma e orçamento gerados, identificando-os com um número de versão.*
Receita	*Armazena as receitas (entradas) realizadas para o projeto.*
PrevisaoReceita	*Armazena as receitas previstas para o projeto.*
Despesa	*Armazena as despesas que podem ocorrer em um projeto.*
DespesaProjeto	*Armazena o valor das despesas de um determinado projeto.*
ValorRecurso	*Armazena o valor dos recursos de um determinado projeto.*
Desvio	*Armazena os desvios de cronograma e orçamento registrados durante o desenvolvimento do projeto.*
CaminhoCritico	*Armazena os caminhos críticos do projeto.*
MarcoPontoControle	*Armazena os marcos e pontos de controle do projeto.*
ConhecimentoEspecialista	*Armazena as dependências usuais entre as atividades.*

Tabela A5.2 – Classes para o Planejamento de Custos (continuação)

A5.3 Diagramas de Classes

As figuras A5.2 a A5.5 representam o modelo de classes contendo as classes acima descritas. O modelo foi dividido em quatro partes para facilitar o entendimento:

- A figura A5.2 apresenta as classes que foram criadas para realizar a Análise de Pontos de Função.
- A figura A5.3 apresenta as classes que foram criadas para realizar o COCOMO II.
- A figura A5.4 apresenta as classes que foram criadas para realizar o planejamento de custos.
- A figura A5.5 apresenta as classes que foram criadas para realizar o planejamento de tempo.

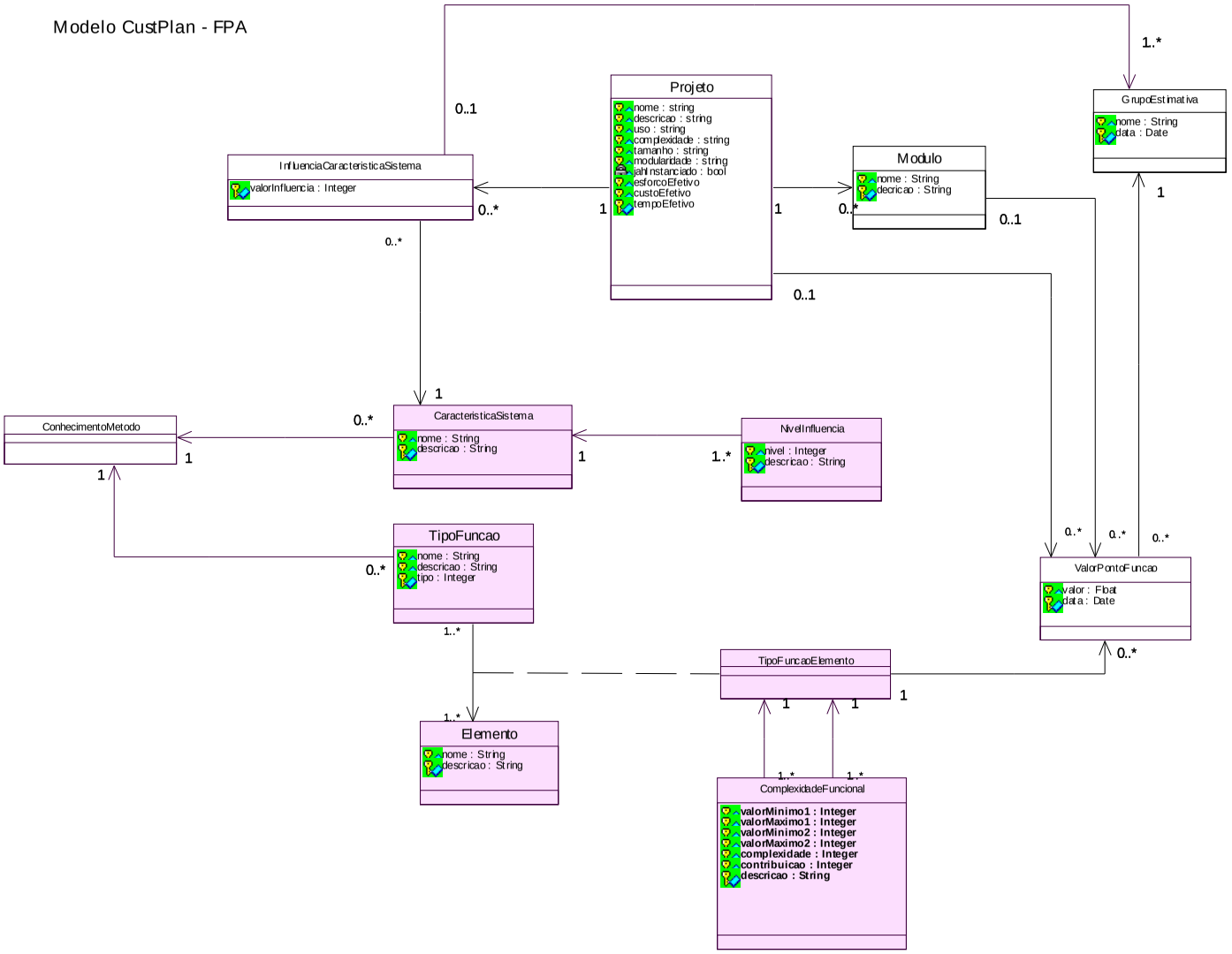


Figura A5.2 – Modelo de Classes – Análise de Pontos de Função

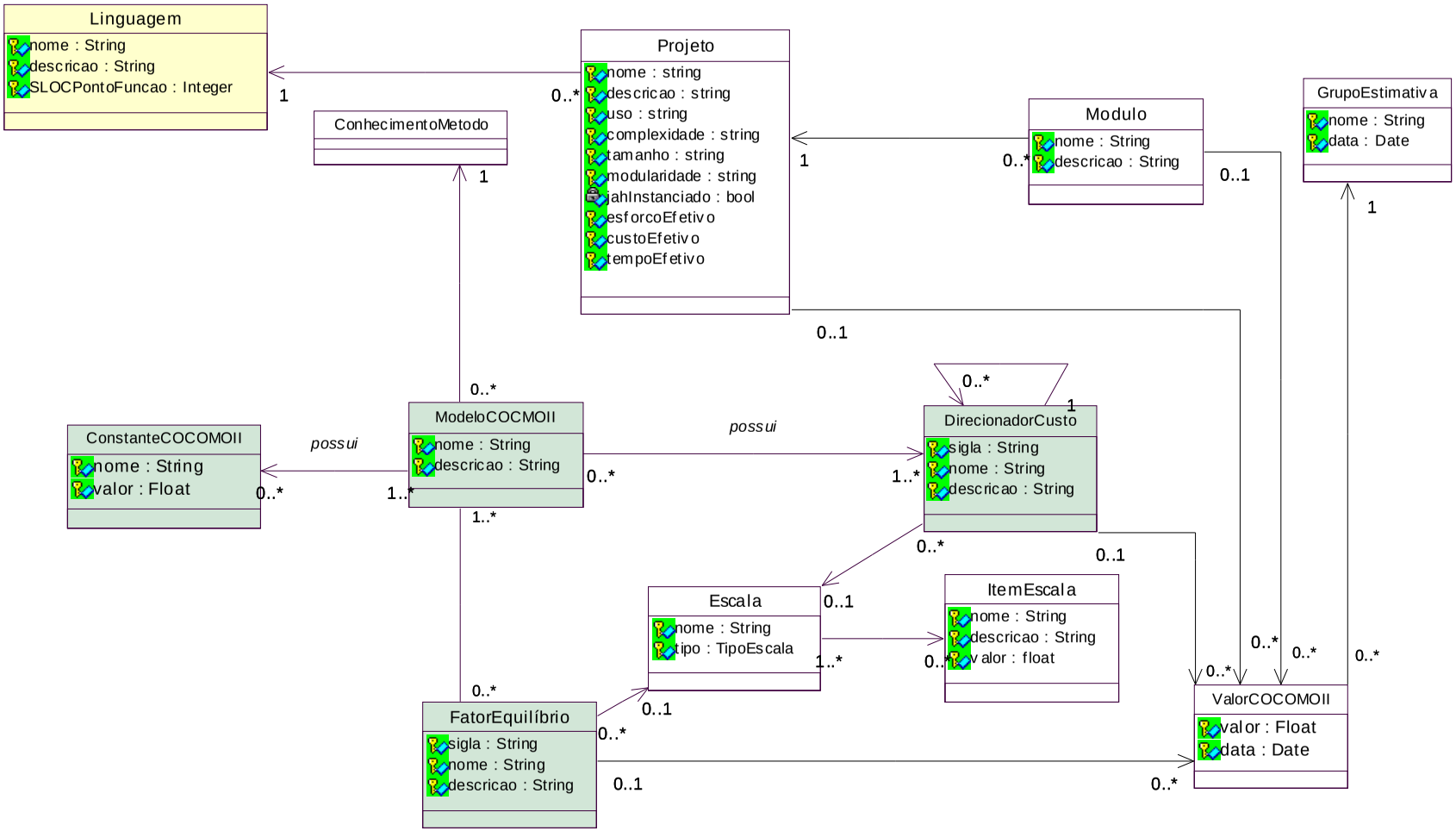


Figura A5.3 – Modelo de Classes – COCOMO II

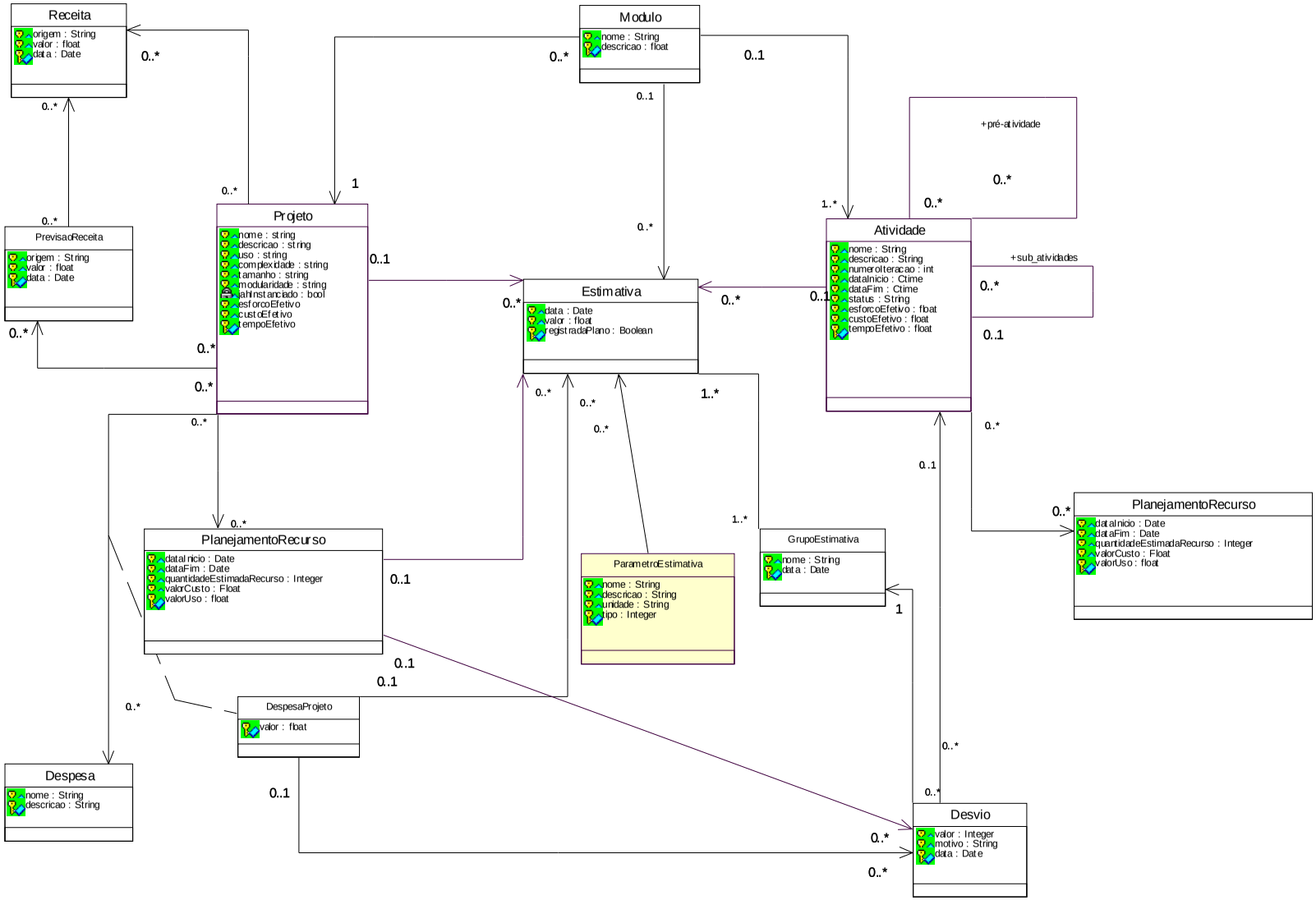


Figura A5.4 – Modelo de Classes – Custos

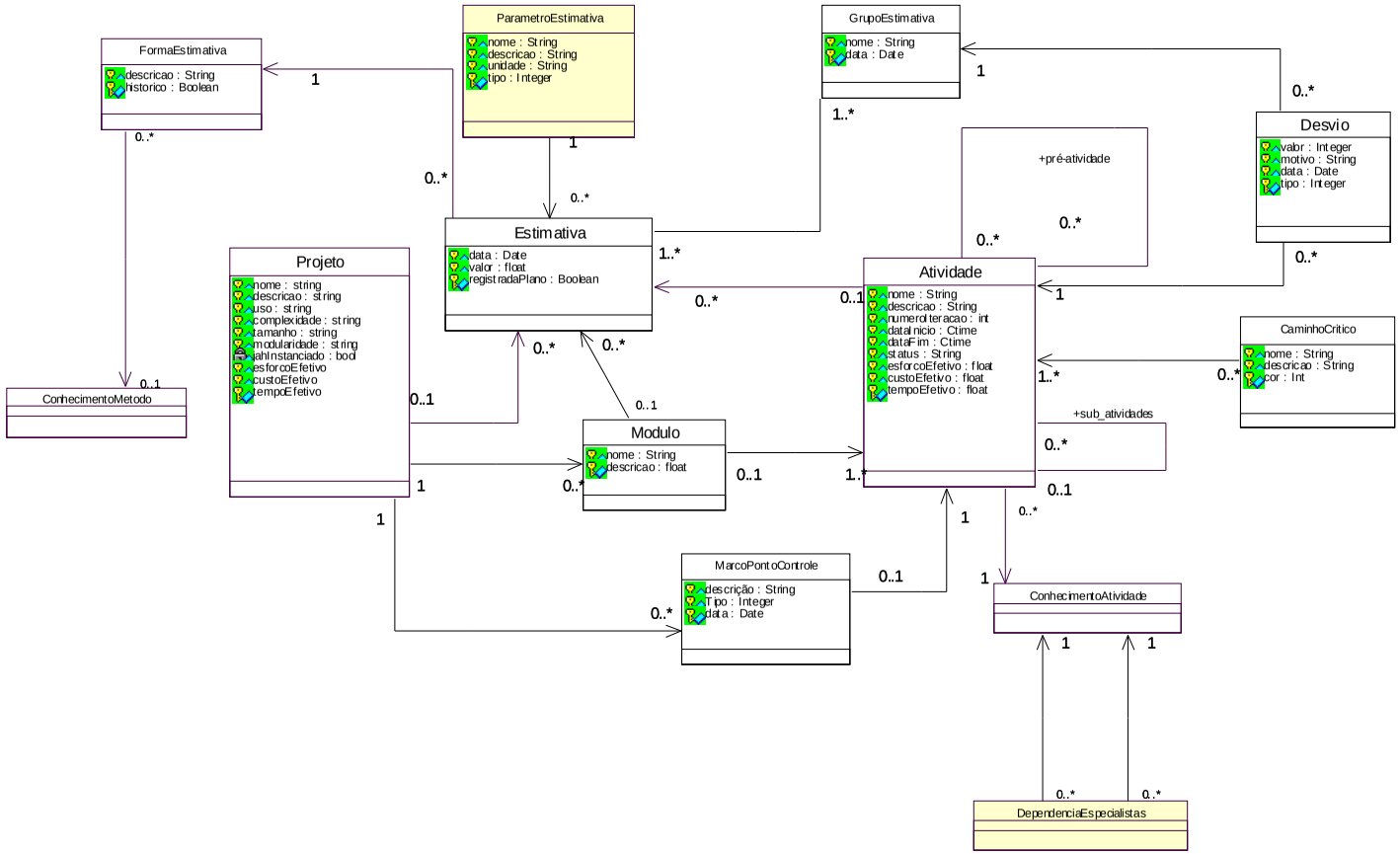


Figura A5.5 – Modelo de Classes – Tempo