FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA – UNIFOR

Yara Maria Almeida Freire

TUCP-M – Pontos de Casos de Uso Técnicos para Manutenção de Software

Fortaleza
2008

FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA – UNIFOR

Yara Maria Almeida Freire

TUCP-M – Pontos de Casos de Uso Técnicos para Manutenção de Software

Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Universidade de Fortaleza como parte dos requisitos necessários para a obtenção do Título de Mestre em Informática Aplicada.

Orientador: Prof. Dr. Arnaldo Dias Belchior

Fortaleza
2008

F866t Freire, Yara Maria Almeida.
TUCP-M – pontos de casos de uso técnicos para manutenção de software / Yara Maria Almeida Freire. - 2008.
154 f.

Cópia de computador.
Dissertação (mestrado) – Universidade de Fortaleza, 2008.
"Orientação : Prof. Dr. Arnaldo Dias Belchior."

1. Software. 2. Tecnologia da informação. 3. Engenharia de software. I. Título.

CDU 681.3.06

Yara Maria Almeida Freire

TUCP-M – Pontos de Casos de Uso Técnicos para Manutenção de Software

Data de Aprovação: 30/06/2008

Banca Examinadora:

Orientador: Prof. Arnaldo Dias Belchior, D.Sc (*In Memorian*)

_____

Prof. Pedro Porfírio Muniz Farias, Dr.
(Presidente da Banca)

_____

Prof. Leonardo Gresta Paulino Murta, Dr.
(Universidade Federal do Rio de Janeiro - UFRJ)

_____

Prof. Nabor das Chagas Mendonça, Ph.D.
(Universidade de Fortaleza - UNIFOR)

Dedicatória

Às minhas filhas Beatriz e Sofia, por serem a alegria, o orgulho e o amor maior da minha vida e à minha irmã querida Yana, de quem eu sinto muita saudade.

Agradecimentos

A Deus, por estar sempre presente em minha vida me confortando nos momentos difíceis, me ajudando a superar os problemas, me dando forças para seguir sempre em frente e me proporcionando momentos de felicidade.

Aos meus pais, José Cândido e Marlise, pelo amor e carinho que sempre me dedicaram. Pelos incontáveis exemplos de perseverança e força diante das dificuldades.

Às minhas filhas, por serem tão maravilhosas e me darem a tranqüilidade necessária para produzir esse trabalho.

À minha irmã Yana (*in memorian*), pela sua eterna torcida e apoio.

À amiga Lívia Nojoza, por suas importantes considerações e críticas a esse trabalho.

À amiga Joy Patrícia, pela utilização e disseminação da técnica aqui apresentada.

Ao professor Dr. Porfírio, por aceitar ser co-orientador desse trabalho.

Aos professores Dr. Nabor das Chagas Mendonça e Dr. Leonardo Murta, pela presença na banca examinadora e por suas importantes considerações que resultaram no aperfeiçoamento desse trabalho.

Ao professor Dr. Arnaldo Belchior (*in memorian*), meu profundo agradecimento e sincera homenagem ao excelente profissional, orientador, pessoa e amigo. Pelo cuidado e consideração dedicados a cada um de seus alunos, pela seriedade e competência com que assumia seu trabalho, por todos os conhecimentos compartilhados.

Resumo da Dissertação apresentada ao MIA/UNIFOR como parte dos requisitos necessários para a obtenção do título de Mestre em Informática Aplicada

TUCP-M - PONTOS DE CASOS DE USO TÉCNICOS PARA MANUTENÇÃO DE SOFTWARE

Yara Maria Almeida Freire

Junho / 2008

Orientador: Dr. Arnaldo Dias Belchior
Co-Orientador: Dr. Pedro Porfírio Muniz
Programa: Informática Aplicada

Considera-se a manutenção de software como a fase iniciada logo após a implantação do produto de software. Pesquisas evidenciam que a quantidade de projetos de manutenção de software nas organizações é bem superior aos projetos de desenvolvimento de novos aplicativos, exigindo maior esforço e custo, particularmente relacionados a software críticos para as organizações. Em virtude desse diagnóstico, é fundamental estimar adequadamente projetos de software dessa natureza, objetivando um melhor planejamento e monitoramento. Diversas técnicas foram propostas e utilizadas para calcular as estimativas de projetos de software. Entretanto, a maioria das técnicas existentes é mais adequada para projetos de desenvolvimento de software ou não tratam todas as peculiaridades da manutenção de software. Este trabalho propõe uma técnica para calcular estimativas de projetos de manutenção de software, estendendo a técnica TUCP (Pontos de Caso de Uso Técnico), utilizada em estimativas de projetos de desenvolvimento de software, baseada em casos de uso. A **TUCP-M (Pontos de Caso de Uso Técnico para Manutenção de Software)** permite um cálculo mais acurado, e de forma simples, para a estimativa de tamanho, esforço, prazo e custo de projetos de manutenção de software.

Abstract of Thesis presented to MIA/UNIFOR as a partial fulfillment of the requirements for the degree of Master of Applied Informatics

TUCP-M - TECHNICAL USE CASE POINTS FOR SOFTWARE MAINTENANCE

Yara Maria Almeida Freire

June / 2008

Advisor: Dr. Arnaldo Dias Belchior

Co-advisor: Dr. Pedro Porfírio Muniz

Program: Applied Informatics

Software maintenance is considered the phase that begins right after the implantation of the software product. Research shows the quantity of software maintenance projects in organizations to be much higher than development projects for new applications, requiring greater effort and expenditures, particularly related to critical software to the organizations. In virtue of this diagnosis, it is fundamental to adequately estimate software projects of this nature, with better planning and monitoring. A variety of techniques were proposed and used to calculate software project estimates. However, most of the existing techniques are more adequate for software development projects or do not deal with the peculiarities of software maintenance. This work proposes a technique that calculates estimates for software maintenance projects, using the TUCP (Technical Use Case Points), used in estimates of software development projects, based on use cases. **TUCP-M (Technical Use Case Points for Software Maintenance)** allows for a more accurate and simple calculation for estimating size, effort, schedule and costs of software maintenance projects.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1
Introdução

Este capítulo apresenta as principais questões que motivaram a realização deste trabalho, seus objetivos e sua organização.

1.1 Motivação

Diante do aumento da competitividade das organizações, onde seus clientes esperam agilidade, segurança e bom atendimento, apresenta-se como inegável a importância da Tecnologia da Informação para que essas organizações alcancem o sucesso almejado. E para manterem-se competitivas no mercado globalizado de hoje, as organizações dão importância cada vez maior a seus projetos de software, que são o suporte para que seus negócios sejam realizados.

Projetos de software nas organizações significam investimento, planejamento dos custos e acompanhamento dos cronogramas, visando atender com qualidade, em tempo hábil e de forma controlada as necessidades que se apresentem. Nesse cenário, torna-se imprescindível estimar o mais cedo possível o esforço e o custo de cada um desses projetos.

O tamanho de um projeto de software é uma das primeiras estimativas a ser realizada, pois dessa dimensão depende a definição de esforço, custo e prazo do projeto. Além de subsidiar o planejamento do projeto, a estimativa de tamanho facilita o relacionamento entre cliente e fornecedor, permite o gerenciamento de riscos, o controle do cronograma e possibilita o conhecimento da produtividade da equipe, o que também beneficia a gerência e a qualidade dos contratos de terceirização de projetos de software (GARMUS; HERRON, 2000 e LONGSTREET, 2002).

Várias técnicas para estimar o tamanho de um projeto de software têm sido propostas e melhoradas desde o final da década de 1960. Dentre elas, a APF (*Function Point Analysis*) (FPCPM, 1999) é uma das mais utilizadas. Segundo Ramil e Lehman (2000), a técnica de Análise de Pontos por Função é uma das mais conhecidas e utilizadas para se estimar o tamanho de projetos de software. No entanto, a APF calcula o tamanho da manutenção de um software como sendo do mesmo tamanho para desenvolvê-lo. Isto eleva o cálculo da estimativa de tamanho da manutenção, tornando-a irreal. (FREIRE & BELCHIOR, 2007c).

Com o aumento do uso da tecnologia Orientada a Objetos (OO) para o desenvolvimento de projetos de software, foi proposta por Karner (1993), para estimar projetos de software orientado a objetos, uma técnica que calcula o tamanho do software através de Pontos de Casos de Uso (PCU).

A principal diferença entre APF e PCU está relacionada ao momento da coleta de informações para a estimativa de tamanho inicial do projeto. A APF possui resultados melhores, na medida em que se tem mais informação da análise e do projeto de sistemas (tabelas, campos, associações, etc.). O PCU tem como proposta ser utilizado logo no início do ciclo de desenvolvimento, na fase de definição dos requisitos, com base no modelo de casos de uso. O modelo de casos de uso é uma técnica largamente utilizada na indústria para capturar e descrever os requisitos funcionais de um software. Esse modelo consiste de diagramas e descrições de casos de uso (ANDA, 2002).

Importante ressaltar que a maioria dos projetos de software nas organizações refere-se à manutenção em seus sistemas já existentes. O que não é difícil de se compreender, pois, se o software atual funciona adequadamente, supõe-se que a organização terá uma preferência muito maior em aplicar a eles apenas os ajustes necessários em função de mudanças nos negócios ou outras necessidades de correções, do que substituí-los. Além disso, não é de se esperar que uma organização de grande porte troque todos seus sistemas somente pelo fato de que a tecnologia neles empregada está ultrapassada. Esses sistemas representam ativos importantes da organização e ela estará disposta a investir de maneira a manter o valor desses ativos (SOMMERVILLE, 2007).

Diante dessa situação, é inegável a importância dos projetos de manutenção de software nas organizações. Corroborando com isso, estatísticas revelaram no início dos anos noventa, que muitas organizações alocaram, no mínimo, 50% de seus recursos financeiros em manutenção de software (APRIL, 1995; BOURQUE *et al.,* 1996). Na década atual, a manutenção de software tem chegado a 70% dos custos de um sistema de software, tornando-se um dos grandes desafios da engenharia de software (WEBSTER *et al*., 2004).

Apesar das pesquisas e das evidências apresentadas no dia-a-dia das organizações, a maioria das técnicas e métodos para se estimar projetos de software privilegiam o desenvolvimento e não a manutenção dos aplicativos, na medida em que não se adaptam adequadamente a todas as peculiaridades das várias categorias nas quais a manutenção de software é classificada. Por exemplo, algumas delas são aplicáveis apenas às manutenções evolutivas, não sendo aplicáveis às manutenções corretivas.

Em 2005, foi apresentada uma extensão da UCP, a TUCP (*Technical Use Case Point*), propondo um cálculo mais acurado para o esforço de projetos, e permitindo uma visão mais detalhada das estimativas por etapa do ciclo de vida do software, possibilitando a realização de refinamentos dessas estimativas para um acompanhamento mais efetivo do projeto (MONTEIRO, 2005). Porém, a técnica não tratava adequadamente projetos de manutenção. Persistia a necessidade identificada por vários autores: apesar de existirem em maior número nas organizações e demandarem um maior custo, os projetos de manutenção de software ainda carecem de métodos e técnicas que permitam obter estimativas mais acuradas.

Como resultado deste diagnóstico, e ainda pela necessidade de se obter uma estimativa na fase de definição dos requisitos do projeto, este trabalho apresenta uma extensão da TUCP, a TUCP-M (Ponto de Caso de Uso Técnico para Manutenção), propondo um cálculo mais acurado para a estimativa de projetos de manutenção de software.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é apresentar uma técnica para estimar projetos de manutenção de software com base nos requisitos, a **TUCP-M** (Pontos de Caso de Uso Técnico para Manutenção de Software), permitindo um cálculo mais acurado e de forma simples, para a estimativa de tamanho, esforço, prazo e custo de projetos dessa natureza.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Considerar no cálculo do tamanho do projeto de manutenção apenas as transações que foram incluídas, alteradas ou excluídas em virtude da solicitação de mudança.
- Propor o fator de manutenção que objetiva refinar o cálculo do esforço necessário a cada etapa do ciclo de vida do projeto de manutenção de software.
- Propor uma fórmula de cálculo da produtividade baseada nos fatores ambientais, com o intuito de diminuir o grau de subjetividade embutido na determinação dessa produtividade.

- Utilizar a TUCP-M como técnica para estimar o tamanho de projetos de manutenção de software em contratos de terceirização.
- Calcular percentual de erro relativo como forma de avaliação e aprimoramento da técnica.
- Contribuir efetivamente no planejamento e execução de projetos de manutenção de software nas organizações.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

No capítulo 2, **Manutenção de Software,** são apresentados os principais conceitos sobre este tema, discorre-se como uma manutenção de software é classificada, do que consiste o processo de manutenção de software e apresenta ainda os principais problemas apontados nesta atividade.

No capítulo 3, **Estimativa de Software,** abordam-se conceitos relacionados à estimativa de projetos de software, destacando-se algumas técnicas utilizadas para estimar o tamanho, esforço, prazo e custo desses projetos e a importância das estimativas para projetos de manutenção de software.

No capítulo 4, **TUCP-M – Ponto de Caso de Uso Técnico para Manutenção de Software,** descreve-se a técnica TUCP-M proposta, expondo ainda um exemplo sobre sua aplicação.

No capítulo 5, **Estudo de Casos,** comenta-se a aplicação da TUCP-M em projetos reais de manutenção de software de uma organização de grande porte. Mencionam-se ainda as percepções e os impactos decorrentes do uso da TUCP-M nessa organização.

No capítulo 6, **Conclusão,** encontram-se os principais resultados da aplicação da TUCP-M.

No Apêndice A, apresenta-se o modelo do documento de solicitação de mudança.

No Apêndice B, mencionam-se algumas sugestões para a contagem das transações.

No apêndice C, mostra-se um exemplo de especificação de caso de uso, identificando as transações de manutenção.

No apêndice D, apresenta-se um documento de solicitação de mudança preenchido.

Capítulo 2

Manutenção de Software

Este capítulo aborda aspectos relacionados à manutenção de software, seus principais conceitos, os diversos problemas envolvidos no tema, os fatores referentes aos custos de manutenção de um sistema e algumas indicações de melhores práticas sobre o tema.

O software não é um produto acabado, está sempre em mutação. Esta realidade tem origem na necessidade de melhoria do processo, produto ou serviço que utiliza a tecnologia da informação na disponibilização de seu uso ou nas mudanças das regras de negócio que regem as organizações. Diante desse fato, a manutenção de software tem grande destaque dentro das organizações que dependem de software para apoiar seus negócios.

2.1 Conceitos e Características

A manutenção de software é conhecida como uma atividade durante a qual ocorrem modificações em um ou mais artefatos construídos no desenvolvimento de um software, buscando mantê-lo disponível, corrigir suas falhas, melhorar seu desempenho e adequá-lo aos requisitos novos ou modificados, conforme as necessidades de seus usuários. Na manutenção de um software pode-se ainda incluir ou excluir alguns artefatos, de acordo com a necessidade apresentada. A manutenção de software é, portanto, caracterizada por qualquer modificação realizada no software após sua entrega. A manutenção é uma atividade inevitável e pode vir a ser a mais longa fase no ciclo de vida do sistema (PIGOSKY, 1996).

Na Engenharia de Software, a manutenção é apresentada como um processo iniciado imediatamente após a implantação do software (CAPELLI *et al*., 2003). É definida pelo IEEE (1998) como a modificação de um produto de software depois de sua entrega ao cliente, para corrigir falhas, melhorar sua performance ou qualquer outro atributo, ou para adaptar o produto a um ambiente modificado. Este processo normalmente é desencadeado por uma solicitação do cliente ou por algum relatório de problemas gerado pelo usuário, sendo que este tipo de solicitação é identificado pelo termo genérico de Requisição de Modificação (IEEE, 1998) e (ISO/IEC 14764, 1999). A manutenção é uma realidade no mercado de software, sendo considerada em estimativas como até 70% de todo trabalho de engenharia de software (GRUBB & TAKANG, 2003).

O esforço despendido para a manutenção do volume de software existente e considerado como legado, ou seja, em funcionamento nas organizações, é relativamente maior que o esforço de desenvolvimento de novos projetos. Normalmente, o conjunto de sistemas mais antigos das organizações contém as funções essenciais do negócio e devem funcionar adequadamente. Sua substituição implica normalmente em um risco maior e um maior custo para a organização. Por isso, as organizações optam por realizar manutenções de seus sistemas mais críticos em vez de simplesmente substituí-los.

Certamente, manutenção em um software é bem mais do que consertar defeitos. Por essa razão, o termo evolução às vezes traz uma melhor caracterização desta atividade, pois o mesmo proporciona uma visão de extensão do sistema, mostrando que tudo que vai além dos requisitos iniciais pode ser considerado uma manutenção. Segundo Pfleeger (2007) existem vários fatores que justificam a necessidade de manter um produto de software, entre eles: dar continuidade ao serviço executado pelo produto, suportar alterações obrigatórias (legais), suportar as melhorias necessárias em termos de funcionalidades do produto e para facilitar futuras manutenções.

Segundo Pressman (2000), a manutenção de software é classificada em quatro categorias: Manutenção Corretiva, Manutenção Adaptativa, Manutenção Perfectiva e Manutenção Preventiva.

- **Manutenção Corretiva:** identificação e remoção de falhas no software (HILBURN *et al*., 1999). Essas falhas podem ser provenientes do desenvolvimento ou de outras manutenções anteriores.
- **Manutenção Adaptativa:** mudanças no ambiente tecnológico onde o software atua. HILBURN *et al*. (1999) consideram uma nova plataforma de hardware, um novo sistema operacional, um novo sistema gerenciador de banco de dados como exemplos deste tipo de mudança.
- **Manutenção Perfectiva:** implementação de novos requisitos funcionais e não funcionais solicitados pelo cliente. Os sistemas de software devem mudar e se adaptar às novidades externas, ou tornam-se progressivamente menos úteis (LEHMAN, 1996).
- **Manutenção Preventiva:** modificação realizada para melhorar a confiabilidade ou a manutenibilidade futura (PFLEEGER, 2007). A manutenção preventiva é executada frequentemente nos produtos de software onde a segurança é crítica (SWEBOK, 2004).

Durante a realização de um projeto de manutenção de software, dificilmente as atividades realizadas serão exclusivamente classificadas como pertencentes a apenas uma das categorias mencionadas. Por exemplo, a manutenção corretiva pode também requerer adições de novas funcionalidades (manutenção perfectiva) em um subsistema. Similarmente, um subsistema pode ser re-projetado para melhorar sua manutenibilidade (manutenção preventiva), ao mesmo tempo em que é modificado para executar em uma nova plataforma de software.

A ISO/IEC 14764 (1999) classifica a manutenção adaptativa e perfectiva como uma melhoria. Agrupa também as manutenções corretivas e preventivas em uma categoria de correção, como mostrado na Tabela 2.1:

Tabela 2.1 – Categorias da Manutenção de Software (SWEBOK, 2004)

	Correção	**Melhoria**
Proativa	Preventiva	Perfectiva
Reativa	Corretiva	Adaptativa

A manutenção pode ainda ser caracterizada ou subclassificada de acordo com seu tamanho e prazo de entrega (CAPPELLI *et al.*, 2003):

- **Tamanho:**

  Grande / Média: Manutenção que envolve mais do que 40 h.h. (homens-hora) de esforço para sua realização.

  Pequena: Manutenção que envolve menos do que 40 h.h. (homens-hora) de esforço para sua realização.

- **Prazo de entrega:**

  Agendável**:** Manutenção necessária, porém para a qual não existe uma data final pré-determinada, ou seja, a mesma pode ser planejada sem que previamente já se tenha uma data para o término da mesma;

  Emergencial: Manutenção para a qual se tem uma data final pré-determinada, mesmo antes de ser feito um levantamento inicial do seu escopo.

As classificações referentes a tamanho e prazo de entrega podem ser combinadas, onde manutenções agendáveis podem ser grandes/médias ou pequenas, bem como manutenções emergenciais também podem ser de grande/médio porte ou pequenas.

Segundo Sommerville (2007), é difícil encontrar números atualizados para o esforço relativo dedicado aos diferentes tipos de manutenção. A Figura 2.1 mostra a noção desse

esforço. Conforme mostrado, o esforço maior de manutenção de software é direcionado para a adição ou modificação de funcionalidades e não para a correção de defeitos.

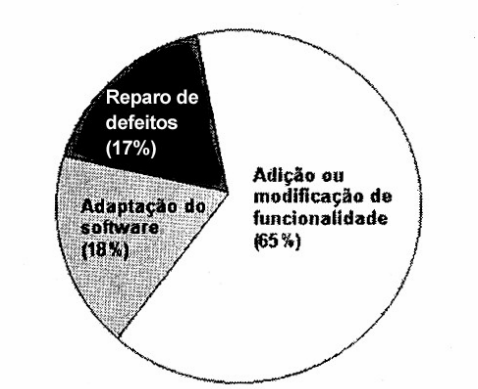


Figura 2.1 – Distribuição do Esforço de Manutenção - Software Engineering (SOMMERVILLE, 2007)

Dados um pouco diferenciados dos apresentados por Sommerville (2007) são apresentados em Pfleeger (2007), embora ambos demonstrem que o reparo de defeitos, caracterizado pela manutenção corretiva, tem uma porcentagem significativa para este tipo de manutenção. Nota-se que a manutenção perfectiva é o tipo com o maior percentual, ou seja, a evolução do software é o foco principal e que a soma dos demais tipos de manutenção totalizam os 50% de esforço restante. A Figura 2.2 apresenta estes dados.

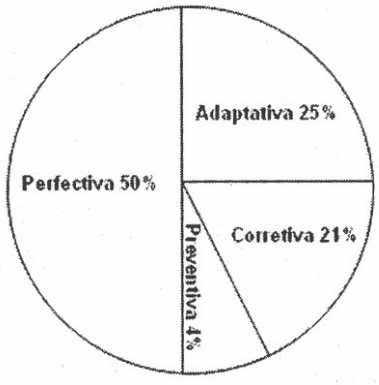


Figura 2.2 – Percentuais de Esforço por Tipo de Manutenção (PFLEEGER, 2007)

Segundo Grubb & Takang (2003), apesar de a manutenção ser considerada parte do ciclo de desenvolvimento do software, existem diferenças fundamentais entre a atividade de desenvolver e a atividade de manter um software. Um novo desenvolvimento objetiva construir produtos novos e a manutenção trabalha com parâmetros e construção de algo sobre um sistema existente. Na manutenção há a necessidade de se ter um conhecimento geral do

que o software faz, como está implementado, identificar onde as mudanças devem ser introduzidas e analisar todos os seus impactos.

Embora muitas atividades relacionadas à manutenção e ao desenvolvimento de produtos de software sejam similares, a manutenção possui características próprias, como detalhadas a seguir (CAPRETZ & CAPRETZ, 1996):

- A manutenção é executada em um produto de software existente. Todas as mudanças introduzidas devem estar de acordo ou ser compatíveis com a sua arquitetura, o seu projeto e código-fonte.
- A manutenção requer tipicamente que programadores gastem uma proporção significativa de seu tempo em tentar compreender como um produto de software é construído e como funciona.
- A manutenção é geralmente ilimitada, continuando por muitos anos, enquanto seja economicamente viável, em contraste com o desenvolvimento, que é comprometido a uma escala de tempo e a um orçamento.
- Durante o desenvolvimento, dados de teste são criados. A manutenção pode usar estes dados e executar testes de regressão, ou, alternativamente, criar dados novos para testar adequadamente somente as mudanças e seus impactos no produto de software.

A manutenção é entendida como uma continuação natural do processo do desenvolvimento de sistema, com atividades associadas de especificação, de projeto, de execução e teste. Um modelo espiral, tal como mostrado na Figura 2.3, é uma representação melhor do processo de software do que representações tais como o modelo de cascata, onde a manutenção é representada como uma atividade do processo separada.

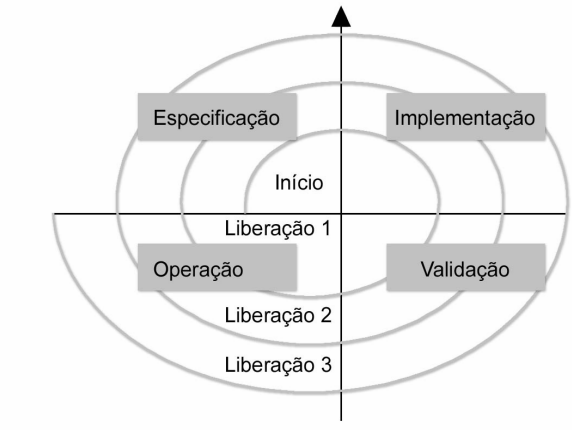


Figura 2.3 – Modelo em Espiral da Manutenção (SOMMERVILLE, 2007)

Inicia-se com a criação de uma primeira versão. Depois da primeira entrega, novos requisitos são propostos e o desenvolvimento da versão 2 é iniciado. A necessidade de evolução apresenta-se óbvia, mesmo antes do sistema ser entregue. (SOMMERVILLE, 2007). Na Seção 2.2 comenta-se mais detalhadamente sobre o processo de manutenção.

2.2 Processo de Manutenção de Software

Um processo de manutenção diz respeito a um conjunto de etapas bem definidas, que direcionam a atividade de manutenção de software, com o objetivo primordial de satisfazer as necessidades dos usuários de maneira planejada e controlada (PIGOSKY, 1996).

Segundo Sommerville (2007), o processo de manutenção varia consideravelmente dependendo do tipo de software a ser mantido, do processo de desenvolvimento usado na organização e das pessoas envolvidas no processo. Em algumas organizações, a manutenção pode ser um processo informal. A maioria dos requisitos da manutenção é repassada na conversa entre usuários e desenvolvedores. Em outras companhias, o processo é formalizado, com documentação estruturada produzida em cada etapa do processo. Todavia, de uma maneira geral, todos os processos de manutenção têm as mesmas atividades fundamentais de análise da mudança, planejamento da nova versão, implementação e entrega da versão ao cliente.

Segundo a NBR ISO/IEC 12207 (1998), o objetivo do processo de manutenção de software é modificar um produto de software existente, preservando sua integridade. Esse processo é ativado, quando o produto de software é submetido a alterações, a partir do momento em que se necessitam efetuar modificações no código e na documentação do sistema devido a um problema, adaptação ou necessidade de melhoria. Dessa forma, o processo de manutenção inicia quando alguma solicitação de modificação é feita, ou pelos clientes, ou pelos próprios mantenedores e termina com a descontinuação do software.

Caso as mudanças propostas sejam aceitas, uma nova versão do sistema é planejada, considerando as mudanças propostas (correção de defeitos, adaptação e novas funcionalidades). As mudanças são implementadas e validadas e uma nova versão do sistema é entregue. O processo pode ser repetido com o surgimento de um novo conjunto de mudanças. A Figura 2.4 mostra um resumo desse processo.

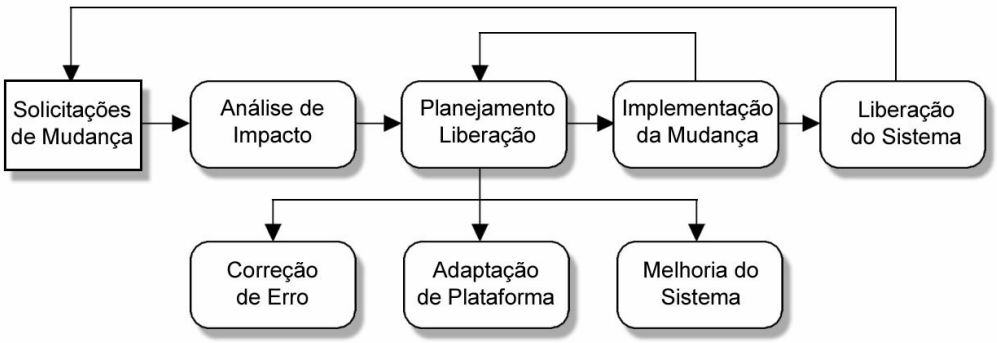


Figura 2.4 – Processo de Manutenção de Software (SOMMERVILLE, 2007)

Pfleeger (2007) ilustra as atividades de manutenção de software, conforme Figura 2.5. Nela, as setas rotuladas na parte inferior representam medições que fornecem informações que os gerentes podem utilizar para decidir quando e como fazer uma alteração.

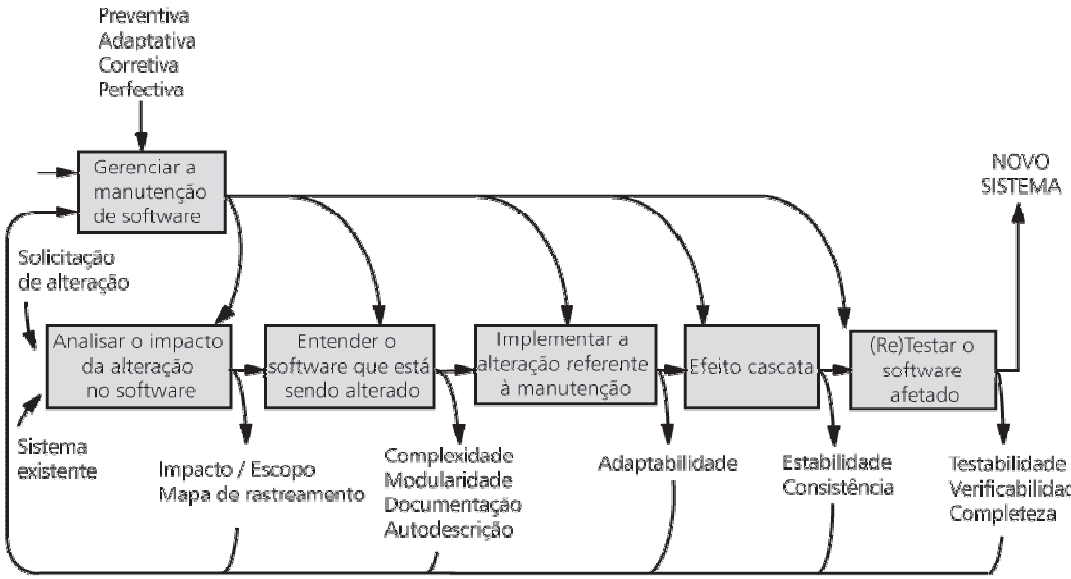


Figura 2.5 – Atividades da Manutenção de Software (PFLEEGER, 2007)

Baseado na NBR ISO/IEC 12207 (1998), descreve-se as atividades que compõem um processo de manutenção de software:

- **Implementação do processo**

Nesta etapa, são construídos os planos e definidos os procedimentos para controlar e documentar a atividade de manutenção e os pedidos feitos pelos solicitantes.

a. O mantenedor deverá produzir, documentar e executar os planos e procedimentos para conduzir as atividades e as tarefas do processo de manutenção.

b. O mantenedor deverá estabelecer procedimentos para receber, registrar e rastrear relatórios de problemas e as modificações requeridas pelos usuários, bem como prover *feedback* para os interessados.
c. O mantenedor implementará o processo de gerência de configuração, objetivando gerenciar modificações no software existente.

- **Análise do problema e da modificação**

Nesta etapa, o mantenedor realiza uma verificação da solicitação, objetivando apresentar várias soluções para o problema identificado.

a. O mantenedor analisará os impactos do problema relatado ou da solicitação de mudança na organização, no sistema existente e outros sistemas envolvidos verificando:
    - Tipo – por exemplo, se a manutenção é do tipo corretiva, melhoria, preventiva.
    - Espaço – por exemplo, qual o tamanho da modificação, o esforço, o custo.
    - Criticidade – por exemplo, qual o impacto no desempenho, na segurança ou corretude.
b. O mantenedor reproduzirá ou verificará o problema.
c. Baseado na análise, o mantenedor considerará as opções de implementação da modificação.
d. O mantenedor documentará o problema/modificação solicitada, os resultados da análise e as opções de implementação.
e. O mantenedor obterá a aprovação para a opção selecionada de modificação.

- **Implementação da Modificação**

Nesta etapa, será realizada a documentação e a alteração do produto de software. Deve ser garantida a perfeita execução da solução proposta.

a. O mantenedor determinará quais documentos, unidades de software e versões necessitam ser modificados. Todas as modificações devem ser documentadas.
b. O mantenedor incorporará o processo de desenvolvimento para executar as modificações. As exigências do processo de desenvolvimento serão suplementadas como segue:

- Serão definidos e documentados os critérios de teste e de avaliação para testar e avaliar os artefatos modificados e não modificados (unidades de software, componentes e itens de configuração).
- Será assegurada a completa e correta execução das exigências novas e modificadas. Será assegurado também que as exigências originais não modificadas não sejam afetadas. Os resultados dos testes serão documentados.

- **Revisão / Aceitação da Modificação**

Nesta etapa, deve-se obter junto ao solicitante seu aceite do produto de software alterado, para que o mesmo possa ser liberado.

a. O mantenedor revisará as modificações para garantir a integridade do sistema modificado.
b. O mantenedor obterá a aprovação para a conclusão satisfatória da modificação, conforme especificada.

- **Migração**

Nesta etapa, o produto gerado é colocado no ambiente de produção e uma avaliação deve ser conduzida para confirmar a execução perfeita da alteração.

a. Se um sistema ou produto de software foi migrado de um ambiente operacional para outro, deve-se garantir que qualquer produto de software ou dados produzidos ou modificações realizadas durante a migração estejam de acordo com este Padrão Internacional.
b. Um plano de migração deve ser produzido, documentado e executado. Esse plano deve incluir:
    - Análise de requisitos e definições da migração
    - Desenvolvimento de ferramentas de migração
    - Conversão do produto e dos dados de software
    - Execução da migração
    - Verificação da migração
    - Suporte para o antigo ambiente no futuro
c. Os solicitantes devem ser notificados do plano e das atividades de migração.

d. A execução paralela nos ambientes novo e antigo poderá ser necessária para garantir uma transição mais tranqüila. Durante esse período, os treinamentos necessários deverão ser realizados.
e. Quando a migração programada é realizada, uma notificação será emitida a todos os interessados. A documentação, os registros e o código associados ao ambiente antigo devem ser arquivados.
f. Após a implantação, uma revisão deverá ser realizada para avaliar o impacto da mudança ao novo ambiente. O resultado dessa avaliação deve ser enviado para as autoridades apropriadas.
g. Os dados usados ou associados ao ambiente antigo devem ficar acessíveis de acordo com as exigências de contrato referentes à proteção de dados e auditorias aplicáveis.

- **Descontinuação do Software**

Nesta etapa, o software será descontinuado por solicitação do usuário. Não haverá mais modificações no mesmo.

- Desenvolver e documentar um plano de descontinuação. O plano deverá conter:
  - Suspensão completa ou parcial após determinado período de tempo
  - Arquivamento do software e de sua respectiva documentação
  - Responsabilidade por algum item residual de sustentação
  - Transição para o novo software, se aplicável
  - Acessibilidade de cópias do arquivo de dados
- Os usuários devem ser notificados sobre o plano de descontinuação e suas atividades.
- Executar em paralelo o novo sistema e o sistema que será descontinuado poderá ser necessário para garantir uma transição mais tranqüila. Durante esse período, os treinamentos necessários deverão ser realizados.
- Quando a descontinuação programada é executada, todos os interessados deverão ser notificados. Toda a documentação, registros e códigos associados devem ser arquivados, quando apropriado.
- Os dados usados ou associados ao produto de software descontinuado devem ficar acessíveis.

Segundo Anquetil *et al.* (2004) a manutenção é uma atividade inevitável e pode vir a ser a mais longa fase no ciclo de vida do sistema. Tal fato pode ser justificado pelas leis da evolução do software, as quais são mencionadas na Seção 2.3.

2.3 Leis da Evolução do Software

Para acompanhar o processo de evolução de um software é requisitado um constante processo de manutenção em seus artefatos. É necessário que o software acompanhe as mudanças de requisitos impostas pelo ambiente no qual ele está inserido. Uma falha em acompanhar essas mudanças pode implicar em perda de qualidade por parte do software ou até mesmo no fim de sua vida útil.

O envelhecimento de um software é um processo inevitável. Esse processo de envelhecimento gera uma necessidade constante de evolução por parte de todos os sistemas que necessitam manterem-se ativos por períodos longos de tempo. Para entender o processo evolutivo em questão, é importante entender as oito leis de evolução de software, conhecidas pelo nome de Leis de Lehman (LEHMAN *et al.,* 1998). Elas demonstram a necessidade de um bom conhecimento e gerenciamento do processo de manutenção. A seguir apresenta-se um resumo destas leis:

- 1ª. Lei – Mudança Contínua: explica que o software deve se adequar ao ambiente em que opera. A evolução deve ser feita baseada no retorno dos usuários e seu nível de satisfação, caso haja resistência em se evoluir o software e conseqüentemente adaptá-lo a realidade de seus usuários, seu nível de satisfação cairá com o tempo.

- 2ª. Lei – Complexidade crescente: de uma forma simples o software aumenta sua complexidade à medida que é alterado, a menos que seja feito um trabalho para mantê-la ou diminuí-la. A cada nova mudança, a estrutura original do software tornar-se-á mais fragmentada e o custo de novas mudanças aumentará gradativamente, até o momento em que estes custos não serão mais viáveis. Será então necessário um trabalho de reestruturação do software para que este tenha sua complexidade diminuída.

- 3ª. Lei – Auto-Regulação: o processo de evoluir um software é auto-regulado, usando um sistema de *feedback*. A evolução de um software é implementada por um grupo de técnicos que opera dentro de uma organização maior. Pontos de controle serão estabelecidos pela gerência para garantir que as normas da organização sejam seguidas e

seus objetivos alcançados em todos os níveis. Os controles de retorno positivos e negativos são exemplos de mecanismos de estabilização.

- 4ª. Lei – Conservação da estabilidade organizacional: se a manutenção não estiver comprometida com a evolução, existe uma tendência de degradação. O nível de atividade vai ser decidido pelas necessidades dos usuários e seus retornos, como mostrado na terceira lei, este nível após um período de tempo, tende a se manter constante, e o nível de pessoas trabalhando no software não pode crescer indefinidamente.

- 5ª. Lei – Conservação da familiaridade: chega um determinado momento em que o software terá uma estabilidade. Um fator determinante na evolução de um software é a familiaridade de todos os membros da equipe com os objetivos desta, quanto mais mudanças forem necessárias, maior vai ser a dificuldade de que toda a equipe esteja ciente dos objetivos.

- 6ª. Lei - Crescimento contínuo: o conteúdo funcional de um software é continuamente aumentado para manter a satisfação do cliente. Após o lançamento do software, mudanças serão necessárias para a contínua satisfação do usuário. Estas mudanças podem ser correções de falhas, adições de novas funcionalidades ou melhorias em funções pré-existentes.

- 7ª. Lei – Qualidade decrescente: o software pode ser alterado e deve manter uma série de outros itens antes já existentes. Programas apresentarão qualidade decrescente a menos que sejam rigorosamente mantidos e adaptados às mudanças no ambiente operacional. Mesmo que este software funcione satisfatoriamente por muitos anos, isto não será um indicativo de que continuará funcionando a contento nos anos vindouros. A sétima lei diz que esse nível de incerteza aumentará com o tempo, a não ser que seja feito um esforço para detectar e corrigir as causas desta incerteza. Este esforço evolutivo deve ser contínuo para todas as novas versões do software.

- 8ª. Lei – Sistema de retorno: é um sistema de retroalimentação. O software está sempre mudando em constante *feedback*, tornando o ciclo muito complexo. A vida de um

software é um ciclo bem estabelecido de retorno positivo e negativo dos usuários entre cada versão do software.

2.4 Problemas da Manutenção de Software

Vários são os problemas técnicos e gerenciais enfrentados pela atividade de manutenção de software, cujas principais causas são: a complexidade do domínio do problema, a dificuldade em gerenciar o processo de desenvolvimento e manutenção, a eventual necessidade de flexibilização do software e a tecnologia ultrapassada presente nos sistemas legados, os quais resistem de maneira significativa a modificações ou evoluções (BOOCH, 1994).

Embora identificada como a atividade mais praticada na área de Tecnologia da Informação, a manutenção de software é vista como uma atividade de segunda classe. Sendo assim, alguns problemas são relacionados à manutenção por falta de conhecimento do que realmente é manter (evoluir) um software. Alguns desses problemas tornam-se mitos nessa área. Segundo Pressman (2000) são eles:

- Atividade mal vista – deixada aos funcionários menos competentes, vista como uma punição, passando a idéia de atividade que oferece menos perspectivas de progressão profissional.
- Atividade ignorada/desconsiderada – não é dada a devida importância a esta atividade. Muitos profissionais não analisam que a manutenção é a atividade de maior ocorrência em um software.
- Problemas com a documentação – normalmente a documentação não existe, está desatualizada ou é muito ruim.
- Problemas com as modificações – as modificações realizadas não são documentadas.
- Falta de controle e disciplina nas atividades de desenvolvimento – os problemas nas atividades de desenvolvimento muitas vezes tornam-se problemas para a manutenção de software.
- Dificuldade de entender o que foi criado – muitas vezes é excepcionalmente difícil entender o programa “de outra pessoa” e a “outra pessoa” normalmente não está por perto para explicar.

- Software não é projetado para sofrer mudanças – a equipe executora do projeto de desenvolvimento geralmente não tem a preocupação com a manutenção deste software, pois em muitos casos, não é a mesma equipe que fará a manutenção.

Além desses problemas, KAJKO-MATTSSON *et al*. (2001) mencionam que a manutenção de software é uma atividade difícil principalmente pelos seguintes motivos:

- **Problemas de Diagnóstico** – meses podem ser gastos em processos investigativos para detectar a real causa do problema antes de corrigi-lo.
- **Falta de Documentação** – inexistência ou desatualização de documentos sobre os requisitos, especificações e processos.
- **Conhecimento sobre o Produto** – dificuldade de familiarização com a estrutura e a composição interna dos produtos a serem alterados.
- **Condução de Mudanças no Software** – falta gerência, ou seja, métricas, análises de riscos, acompanhamento, controle e documentação.
- **Habilidades de Escrita** – manter um software não é somente programar, precisa-se saber documentar o que se está programando.
- **Baixa Estima dos Mantenedores** – os profissionais geralmente dão preferência para atuar em novos projetos. Manter um software reduz o moral da equipe e compromete a qualidade e o rendimento da atividade como um todo.

Entre diversos outros problemas como insatisfação do cliente em relação a pedidos de reparo, introdução de defeitos como resultado de mudanças feitas de maneira incorreta e sem documentação, a falta de documentação é o problema que traz uma grande dificuldade para o processo de manutenção, pois a documentação não existe ou não retrata a realidade, ou seja, está desatualizada.

Observa-se ainda que há um problema cultural em engenharia de software na forma de uma visão um tanto quanto tendenciosa, pois vários segmentos do meio acadêmico e da indústria priorizam o ensino e aprendizado do chamado "Desenvolvimento de Software", ou seja, a construção de um novo produto. Os processos propostos são geralmente processos de desenvolvimento ou são processos iterativos que mesclam a manutenção no decorrer de cada iteração (estratégia não tão eficaz quando aplicada em sistemas legados) (RAMOS, *et al*., 2004).

Existem diversas publicações que tratam do assunto "manutenção de software". Dentre elas, destaca-se o SWEBOK, o qual apresenta um conjunto de melhores práticas

relacionadas à engenharia de software. Entre os temas apresentados, encontra-se a manutenção de software. A Seção 2.5 relaciona algumas dessas melhores práticas.

2.5 Guia para a Manutenção de Software

Objetivando reunir em um único documento as melhores práticas referentes à Engenharia de Software, foi construído sob o patrocínio do IEEE o *Guide to the Software Engineering Body of Knowledge*, conhecido pela sigla SWEBOK. O SWEBOK apresenta uma classificação hierárquica dos tópicos tratados pela Engenharia de Software, onde o nível mais alto são as Áreas de Conhecimento. O guia deve necessariamente evoluir conforme a evolução da Engenharia de Software, constituindo-se em elemento valioso de sua infra-estrutura.

O SWEBOK discorre sobre dez áreas de conhecimento: Requisitos de Software, Projeto de Software, Construção de Software, Teste de Software, Manutenção de Software, Gerência de Configuração de Software, Gerência da Engenharia de Software, Processo da Engenharia de Software, Ferramentas e Métodos da Engenharia de Software e Qualidade de Software. A Figura 2.6 foi retirada do SWEBOK (2004) e mostra a estrutura analítica relacionada à área de conhecimento de manutenção de software.

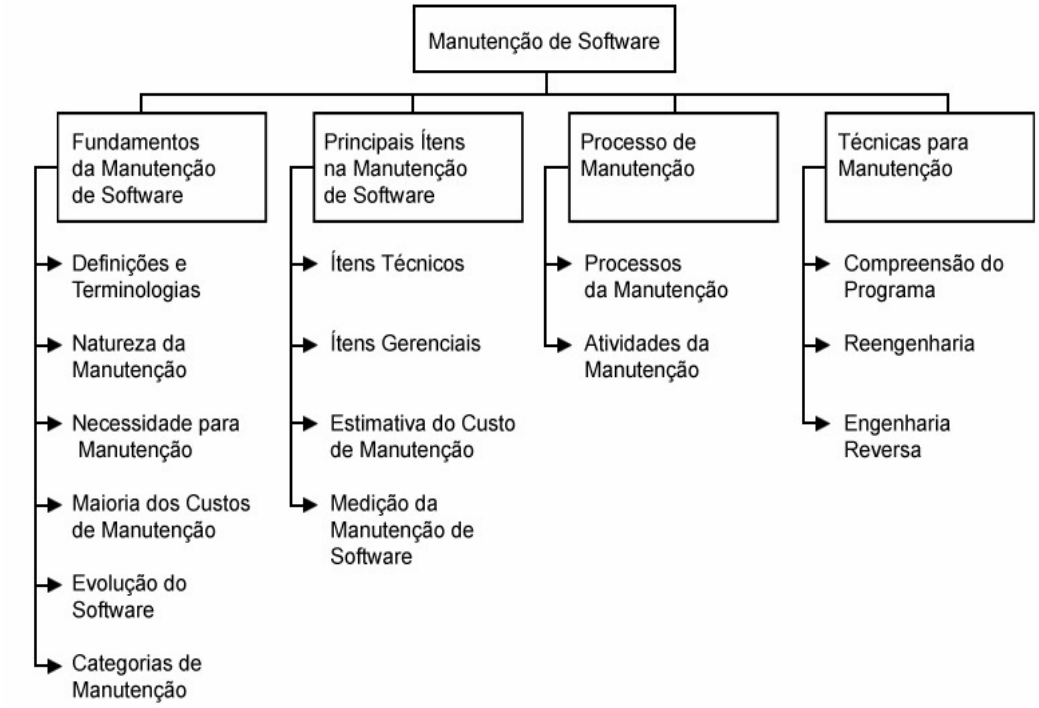


Figura 2.6 – Estrutura analítica da área de conhecimento de manutenção de software (SWEBOK, 2004)

Dentre os assuntos tratados pelo SWEBOK, vale ressaltar os seguintes:

- A manutenção de software produz desafios únicos tanto em nível técnico como em nível gerencial.
- A análise de impacto descreve como conduzir, quanto custa efetivamente, uma mudança em software existente. O objetivo da análise de impacto é:
  - Determinação do escopo da mudança a fim de planejar e executar o trabalho
  - Desenvolvimento de estimativas acuradas dos recursos necessários
  - Análise do custo/benefício da mudança solicitada
  - Comunicação da complexidade da mudança solicitada
- Manutenibilidade: definida pelo IEEE como a facilidade com que o software pode ser mantido, evoluído, adaptado ou corrigido para satisfazer os requisitos estabelecidos. As sub-características da manutenibilidade devem ser especificadas, revistas e controladas durante o desenvolvimento do software, objetivando reduzir o custo da manutenção. Isto é freqüentemente difícil de conseguir porque as sub-características da manutenibilidade não são um foco importante durante o processo de desenvolvimento. Este fato resulta frequentemente na falta de documentação do sistema, o que por sua vez é uma das principais dificuldades na compreensão dos programas e na análise de impacto. Também se observa que a presença de processos sistemáticos e maduros, técnicas e ferramentas ajudam a promover a manutenibilidade de um sistema.
- Alinhamento com os objetivos organizacionais: frequentemente, a manutenção de software objetiva estender a vida útil de um sistema o máximo possível. Além disso, pode também ser solicitada para atender a uma nova funcionalidade demandada pelo usuário. Em ambos os casos, do ponto de vista de um gerente sênior, o retorno do investimento é pouco claro, tornando-se frequentemente uma atividade que consome recursos significativos sem um claro benefício quantitativo para a organização.
- Para os propósitos do planejamento, estimar os custos é um aspecto importante na manutenção de software:
  - Estimativa de custo: as estimativas de custo são afetadas por muitos fatores técnicos e não técnicos. ISO/IEC14764 (1999) indica que “as duas formas mais

populares de estimar recursos para a manutenção de software é o uso de modelos paramétricos e o uso da experiência". Mais frequentemente, uma combinação destes é utilizada.

- Modelos paramétricos: esforços estão sendo empreendidos no intuito de aplicar o modelo de custo paramétrico à manutenção do software. Isto significa que os dados dos projetos passados são necessários para usar nos modelos.
- Experiência: claramente a melhor aproximação nas estimativas de manutenção combina dados e experiência empíricos. Esses dados devem ser fornecidos como resultado de um programa de medição.

Até esse ponto foram apresentados diversos aspectos relacionados ao tema "manutenção de software". Entende-se que o custo de um projeto de manutenção de software é um dos aspectos mais relevantes sobre o tema e, portanto, está referenciado separadamente na Seção 2.6.

2.6 Custo da Manutenção de Software

Koskinen (2003) menciona que no ano de 2000 os custos com a evolução do software, no que diz respeito a manter e evoluir os sistemas, sobre os custos totais do software alcançaram uma proporção maior que 90%. Ainda sobre o assunto, Sommerville (2007) relata que os custos gerais de manutenção ao longo do tempo podem diminuir, à medida que é dedicado um esforço durante o desenvolvimento do sistema, objetivando produzir um sistema com maior facilidade de manutenção. Vale salientar que por mais que um sistema seja produzido visando facilitar sua manutenção, ela sempre existirá, pois adaptações ao meio e aos interessados pelo sistema, bem como acréscimos estarão sempre presentes.

No início dos anos noventa, estatísticas revelaram que muitas organizações alocaram, no mínimo, 50% de seus recursos financeiros em manutenção de software (APRIL, 1995; BOURQUE *et al*., 1996). Na década atual, a manutenção de software tem chegado a 70% dos custos de um sistema de software, tornando-se um dos grandes desafios da engenharia de software (WEBSTER *et al*., 2004).

As diferenças existentes entre o desenvolvimento e a manutenção de software contribuem para o aumento do custo de manutenção. Segundo Sommerville (2007), uma das razões que torna o custo de manutenção elevado é o fato de que é mais caro adicionar novas funcionalidades em um sistema existente do que implementar essa mesma funcionalidade

durante o desenvolvimento de um sistema. Os pontos chaves que distinguem desenvolvimento e manutenção e que elevam o custo de manutenção são (SOMMERVILLE, 2007):

- Estabilidade da equipe – com a finalização do desenvolvimento de um sistema, normalmente a equipe é desfeita e alocada em outros projetos. A nova equipe ou pessoa responsável pela manutenção desse aplicativo não terá domínio sobre ele. Um grande esforço é despendido para obter o entendimento do sistema antes de poder realizar alguma mudança.
- Responsabilidade Contratual – o contrato de manutenção de um sistema é usualmente distinto do contrato de desenvolvimento, inclusive firmados com empresas diferentes. Isso pode significar que a equipe de desenvolvimento não tenha incentivo para desenvolver um sistema fácil de manter. Torna-se mais vantajoso realizar uma entrega mais rápida mesmo que isso implique em um aumento de custo de manutenção.
- Perfil da equipe – freqüentemente, a equipe de manutenção é relativamente inexperiente e sem familiaridade com o aplicativo. Os sistemas foram construídos com linguagens que já estão obsoletas. A equipe não tem experiência com essas linguagens e precisam aprendê-las antes de manter o sistema.
- Estrutura e idade dos programas – à medida que os programas envelhecem sua estrutura tende a ficar degradada, aumentando a dificuldade em entendê-los e mantê-los. Além disso, muitos dos sistemas legados foram desenvolvidos sem a utilização de modernas técnicas de engenharia de software. Não existe documentação ou ela está obsoleta. Os sistemas antigos não estão sob uma gerência de configuração, o que dificulta encontrar a versão correta para ser mantida.

Os custos de manutenção dependem do número de mudanças solicitadas no software e o custo da implementação dessas mudanças depende do grau de manutenibilidade do sistema. Quanto mais complexo o sistema, maior será seu custo de manutenção (SOMMERVILLE, 2007).

Existem dois importantes fatores que impactam diretamente no momento de se decidir por realizar uma manutenção em um sistema: os benefícios que esta mudança trará e o custo que esta mudança terá. Geralmente, os benefícios são rápida e facilmente relacionados. Entretanto, o custo referente a projetos desta natureza é de difícil estimativa.

Sneed (1995) comenta que o custo estimado de uma manutenção deve ser realizado por um método de estimativa de manutenção, rápido e barato. Não se dispõe de tempo para realização de uma longa análise de impacto como em um projeto de desenvolvimento de

software. Isto significa que deve existir um processo pré-definido de estimativa de projetos de manutenção e que este processo esteja apoiado por alguma ferramenta.

É com essa expectativa que esse trabalho propõe a técnica apresentada no Capítulo 4.

2.7 Conclusão

A grande maioria dos sistemas sofre alterações ao longo de todo seu ciclo de vida. Em geral, os sistemas que sofrem manutenção têm seus requisitos originais alterados para atender uma nova legislação, pelo surgimento de novas funcionalidades, pela mudança de hardware ou do ambiente operacional, pela correção de defeitos ou pela prevenção de possíveis falhas (FREIRE & BELCHIOR, 2006).

A evolução do software, a qual é chamada de manutenção, sempre vai ocorrer, sendo necessário criar mecanismos para facilitar e agilizar essa importante atividade. Pfleeger (2007) comenta que a manutenção de software enfoca simultâneamente quatro aspectos principais da evolução dos sistemas:

- Manter o controle sobre as funções do dia-a-dia do sistema
- Manter o controle sobre as modificações do sistema
- Aperfeiçoar as funções aceitáveis já existentes
- Tomar medidas preventivas para que o desempenho do sistema não diminua para níveis inaceitáveis

Na prática o que se observa é que as organizações realizam projetos de manutenção de software com o intuito de manter seus sistemas funcionais e atraentes para os usuários. Este tipo de projeto de software prevalece nas organizações, em especial os que estão relacionados a sistemas responsáveis pelos negócios vitais destas organizações. A substituição destes sistemas implica normalmente em um risco maior e um maior custo para as organizações. Nessa situação, a grande maioria das organizações opta por realizar manutenções em seus sistemas mais críticos em vez de simplesmente substituí-los. Os projetos referentes a estas manutenções são fundamentais, pois garantem a qualidade geral do produto à medida que as necessidades dos usuários/clientes e do ambiente de negócios evoluem.

É necessário perceber a importância da manutenção de software e mudar a cultura de que a mesma não é relevante ou de que deve ser tratada como um “processo de segunda classe”. Ao contrário, a manutenção de software é responsável por aumentar a vida útil dos sistemas mais críticos das organizações.

Nesse contexto, as organizações devem buscar constantemente a melhoria do seu processo de manutenção de software, utilizando metodologias e técnicas adequadas, tratando-o com a importância devida, planejando-o e executando-o de forma eficiente, objetivando a obtenção dos melhores resultados de seus sistemas. Uma falha nesse processo pode significar grandes prejuízos.

No capítulo seguinte, será abordado o tema “Estimativa de Software”, que auxilia no planejamento e gerenciamento mais adequados de projetos de software. Serão comentadas algumas técnicas para calcular estas estimativas e o direcionamento destas técnicas para projetos de manutenção de software.

Capítulo 3

Estimativa de Software

Este capítulo aborda aspectos relacionados à estimativa de projetos de software, destacando algumas técnicas utilizadas para estimar o tamanho, esforço, prazo e custo. Também aborda a importância das estimativas para projetos de manutenção de software.

A engenharia de software recomenda a implantação de atividades de estimativas de tamanho, esforço, prazo e custo como formas de melhorar o planejamento e o acompanhamento de projetos de software. Estimar o esforço necessário para o desenvolvimento de determinado sistema não é uma tarefa trivial. A dificuldade em estimar é especialmente percebida no início do projeto, quando os requisitos ainda não estão totalmente detalhados, muitas vezes não se conhece a equipe que será alocada ao projeto e ainda não se tem certeza das tecnologias a serem utilizadas. Entretanto, é necessário se ter uma estimativa o mais acurada possível, com maior brevidade, para que os negócios de uma organização possam ser adequadamente planejados.

Projetos subestimados provavelmente causarão prejuízos. Por outro lado, projetos superestimados levam a perda de competitividade. Estimativas eficientes permitem a verificação da viabilidade do projeto, a elaboração de propostas técnicas e comerciais, a confecção de planos e cronogramas detalhados, o efetivo acompanhamento dos projetos e até a possibilidade de avaliar a produtividade das equipes que participaram do trabalho. A importância de estimativas confiáveis para projetos de software é consenso entre a comunidade acadêmica e as empresas. Entretanto, é bastante conhecido o fato de que obter estimativas acuradas é um grande desafio. No caso de projetos de manutenção de software, a situação tende a se agravar, pois o conhecimento e as técnicas para se medir um projeto dessa natureza ainda não levam em consideração algumas de suas peculiaridades. Segundo FENTON e PFLEEGER (1997), apesar de não se esperar que as estimativas sejam exatas, elas devem ser precisas para permitir bons níveis de segurança em tomadas de decisão.

A falta de uma técnica para estimar projetos de software e de um processo formalmente definido de avaliação constante entre o que foi previsto e o que foi realizado dificulta uma gerência efetiva desses projetos nas organizações. Planejar, priorizar, negociar e controlar esses projetos sem uma estimativa de tamanho, esforço, prazo e custo é tarefa árdua.

Em um mercado cada vez mais globalizado e competitivo, a eficiência no planejamento de projetos, compreendendo a entrega de produtos de alta qualidade e dentro dos prazos e orçamentos esperados, tem sido um fator crítico de sucesso para o bom desempenho empresarial (SIMÕES, 1999).

As empresas adotam um processo para medir o tamanho dos projetos de desenvolvimento de sistemas com o objetivo de obter subsídios para determinar seu esforço, recursos, duração e custo. De posse dessas informações, que são as estimativas do projeto, é possível apresentar expectativas realistas para a comunidade de usuários, obtendo um relacionamento positivo e de confiança entre as partes. Além disso, as estimativas revelam-se de grande importância no gerenciamento de contratos de terceirização de serviços e avaliação do processo e da equipe de desenvolvimento e manutenção de software utilizados.

Diante dessa realidade, muitos estudos foram realizados na área de estimativas de tamanho de software, resultando na proposição de diversas técnicas e/ou métodos que pudessem auxiliar as equipes no cálculo de estimativas mais acuradas, contribuindo para um planejamento adequado dos projetos e minimizando fracassos em relação à falha de cronograma e extrapolação de custos. Algumas dessas técnicas serão comentadas adiante. Antes disso porém, é necessário discorrer sobre a definição de alguns termos importantes na comunicação entre os profissionais da área.

3.1 Conceitos Básicos

Medida é a indicação quantitativa de um atributo, como o tamanho, a dimensão, a capacidade de um produto ou de um processo (PRESSMAN, 2000). Um exemplo de medida, que é um número bruto (POLLICE, 2004), é o número de defeitos encontrados em um programa.

Medição é o ato de determinar uma medida (POLLICE, 2004), de se medir o número de defeitos no código, por exemplo. Segundo FENTON e PFLEEGER (1997), medição é o processo por meio do qual, números e símbolos são atribuídos do mundo real de forma a tornar possível caracterizar cada entidade utilizando regras claramente definidas, isto é, a medição compreende o processo de obtenção de uma medida para uma entidade do mundo real.

Métrica é uma indicação quantitativa de uma medida para que um sistema ou componente possua um atributo (PRESSMAN, 2000). Assim, uma métrica poderia ser, por

exemplo, a quantidade de defeitos por centena de linhas de código. Em outras palavras, a métrica é descrita em termos de relacionamento entre as medidas (POLLICE, 2004).

As **estimativas** compreendem um processo que se inicia com a medida de tamanho do software para, em seguida, identificar o esforço necessário para a sua construção (AGARWAL *et al.,* 2001). Por sua vez, o esforço estimado é fundamental na elaboração dos cronogramas, gerando, ao final, os prazos e custos do projeto de software (BARCELLOS, 2003).

Potok e Vouk (1999) afirmam que estimar precisamente pode ser a diferença entre o sucesso ou falha de um projeto de desenvolvimento de software. As organizações necessitam estimar o esforço e custo de seus projetos. Para realizar essa atividade, são utilizadas uma ou mais técnicas descritas na Tabela 3.1 (SOMMERVILLE, 2007):

Tabela 3.1 – Técnicas de estimativas de software (SOMMERVILLE, 2007)

Técnica	Descrição
Modelo de Custo Algorítmico	Um modelo é desenvolvido utilizando informações de custo históricas que relaciona algumas medidas de software (normalmente seu tamanho), ao custo do projeto. Uma estimativa é feita a partir da medida e o modelo prediz o esforço exigido.
Julgamento de Especialistas	Vários especialistas nas técnicas de desenvolvimento de software propostas e no domínio da aplicação são consultados. Cada um deles estima o custo do projeto. Estas estimativas são comparadas e discutidas. O processo é repetido até que se chegue a um acordo.
Estimativa por analogia	Esta técnica é aplicável quando outros projetos no mesmo domínio de aplicação são concluídos. O custo de um novo projeto será estimado por analogia com estes projetos concluídos.
Lei de Parkinson	A lei de Parkinson afirma que o trabalho se expande para preencher o tempo disponível. O custo é determinado pelos recursos disponíveis, e não por uma avaliação objetiva. Se o software tem que ser entregue em 12 meses e 5 pessoas estão disponíveis, o esforço requerido é estimado em 60 pessoas-mês.
Preço para ganhar	O custo do software é estimado de acordo com o valor que o cliente tem disponível para aplicar no projeto. A estimativa de esforço depende do orçamento do cliente e não das funcionalidades do software.

3.1.1 Classificação das Métricas

As métricas são classificadas de diversas maneiras de acordo com diversos autores. Algumas dessas classificações são:

- Classificação segundo FENTON e PFLEEGER (1997):
  - Métricas de processo: medem as atividades realizadas durante todo o desenvolvimento do software.
  - Métricas de produto: medem os artefatos, produtos e documentos gerados pelo processo.
  - Métricas de recursos: medem as entidades requeridas para a execução do processo.
- Classificação segundo MÖLLER e PAULISH (1993):
  - Métricas objetivas: podem ser quantificadas e medidas através de expressões numéricas ou representações gráficas de expressões numéricas contadas a partir do código fonte, projeto, dados de teste e outras informações do software.
  - Métricas subjetivas: são medidas baseadas em estimativas pessoais ou de grupo, geralmente obtidas através de conceitos como excelente, bom, regular e ruim. Segundo Fenton e Pfleeger (1997), métricas subjetivas dependem da pessoa que está mensurando, do seu julgamento e do grau de imprecisão, o que dificulta o consenso entre possíveis atributos que envolvam processos, produtos ou qualidade.
- Classificação segundo FENTON e PFLEEGER (1997); e PRESSMAN (2000).
  - Métricas diretas: são aquelas que não dependem da medida de outro atributo, mas da quantificação de um fator observado no produto. Representam tudo aquilo que se pode medir com maior precisão (PRESSMAN, 2000), como por exemplo, o comprimento de um parafuso, o custo e o esforço aplicado no desenvolvimento e na manutenção do software e do produto, linhas de código e velocidade de execução.
  - Métricas indiretas: envolvem as medidas de um ou mais atributos a estes relacionados. São difíceis de ser avaliadas, pois não se consegue facilmente medir, por exemplo, qualidade, eficiência, funcionalidade, complexidade, etc. (PRESSMAN, 2000).

3.1.2 Abordagens de Estimativas

Abordagens de estimativas de software podem ser compostas de modelos matemáticos complexos, expressões aritméticas simples, um conjunto de regras ou uma lista descritiva de

instruções (KOSLOSKI, 2005). Dentro da categoria de métricas diretas, McGarry *et al*. (2002) destacam algumas abordagens de métricas de estimativas de projeto de software como: *i*) modelos paramétricos; *ii*) analogia; *iii*) julgamento de especialistas; *iv*) modelo de estimativas baseado em atividades e *v*) relações simples de estimativas.

As abordagens mais comuns são os modelos paramétricos, analogia e julgamento de especialistas.

Os **modelos paramétricos** assumem que existe um relacionamento matemático entre o tamanho, esforço, cronograma e qualidade e que o relacionamento é afetado por fatores mensuráveis de desempenho também chamados parâmetros. A entrada mais importante nesse modelo é o tamanho, o qual representa a quantidade de funções do software. Para obter melhores resultados, os modelos paramétricos têm que ser calibrados com dados do ambiente local de desenvolvimento (McGARRY *et al*., 2002).

A **analogia** é geralmente realizada por especialista em estimativa (com base na experiência de projetos anteriores) e por algoritmos de busca de projetos similares (disponíveis em base de dados) (JORGENSEN; INDAHL; SJOBERG, 2003). Esta abordagem requer um conhecimento detalhado do projeto para identificar as diferenças específicas entre o projeto proposto e os projetos realizados anteriormente, que foram usados como base para fazer a estimativa. O relatório do estudo realizado por Heemstra (1992), *apud* Jorgensen; Indahl; Sjoberg (2003) em 600 organizações destaca que a estimativa por analogia é o método mais comum de estimativa na indústria de software.

O **julgamento de especialistas** é realizado com base nas experiências de profissionais em projetos semelhantes. As técnicas baseadas na experiência de especialistas são úteis na ausência de dados quantitativos. Longstreet (2002) sugere que ao "avaliar projetos semelhantes, deve-se considerar o tipo de plataforma de hardware, o tipo de linguagem, o tipo de projeto, o tipo de sistema operacional e identificar os dados históricos de taxa de entrega, de cronogramas e de custo do projeto".

O **modelo de estimativas baseado em atividades** segue uma abordagem "*bottom-up*" a partir da avaliação dos tamanhos, esforços, prazos e custos de produtos de trabalho e atividades individuais. As estimativas destes elementos são feitas por opinião de especialistas, ou por comparações com dados históricos ou ainda por meio de combinações entre estas duas formas. Os valores globais da estimativa são obtidos pela agregação dos valores estimados para cada um dos produtos de trabalho analisados (MCGARRY *et al*, 2002).

As **relações simples de estimativas** constituem uma forma simplificada de utilização de modelos paramétricos baseados em dados históricos locais e equações aritméticas, ao invés de modelos matemáticos complexos (MCGARRY *et al.,* 2002). Um exemplo desta forma consiste em estimar o esforço como uma função do tamanho do produto a ser construído e da produtividade esperada para o seu processo de desenvolvimento (IFPUG, 2002). O tamanho pode ser dado em várias unidades tais como: linhas de código ou pontos de função, dentre outras (FENTON e PFLEEGER, 1997).

Para obter resultados significativos, todas as abordagens de métricas de estimativas requerem alguns dados históricos similares ao projeto proposto incluindo, o domínio da aplicação, o nível de maturidade do processo de desenvolvimento, a experiência da equipe, taxas de produtividade, tipos de tecnologia e de ferramentas utilizadas e o grau de reuso (McGARRY *et al*., 2002; FENTON; PFLEEGER, 1997).

Segundo Fenton e Pfleeger (1997), para as principais métricas diretas (analogia, julgamento de especialistas e modelos paramétricos), duas abordagens de estimativas ainda podem ser usadas: *top-down* e *bottom-up*.

As **estimativas *top-down*** (GARMUS; HERRON, 2000; FENTON; PFLEEGER, 1997; KARNER, 1993; McPHEE, 1999; RIBU, 2001) são utilizadas para estimar o projeto, quando informações sobre o escopo do projeto são ainda muito limitadas, pois estimam o projeto como um produto único. Os componentes, as atividades e as fases do projeto são calculados como porções relativas ao todo estimado. A granularidade das estimativas dos produtos de trabalho é determinada com base na estimativa total do projeto (MYLIUS; MARODIN, S.d).

As estimativas *top-down* têm a vantagem de exigir muito menos esforço para elaboração e pouco tempo para serem desenvolvidas, mas como desvantagem sua baixa precisão.

As **estimativas *bottom-up*** (BOEHM, 1981) são obtidas primeiramente através das estimativas das partes do projeto e a estimativa total do projeto é obtida por meio da soma de suas partes. Essa abordagem envolve estimar a atividade ou produto a ser executado no projeto, depois sumarizá-lo ou agregá-lo para obter uma estimativa total do projeto. Referida abordagem solicita que primeiramente sejam levantadas todas as atividades para o projeto, a partir da elaboração de uma WBS (*Work Breakdown Structure*) ou Estrutura Analítica de Trabalho, contendo todas as possíveis atividades a serem realizadas no projeto.

A vantagem desse tipo de estimativa é que ela gera resultados precisos. Sua precisão depende do nível de detalhe que está sendo considerado. Sua desvantagem é a ausência de alguma atividade importante a ser contemplada na WBS. Isso reflete a dificuldade na estimativa do esforço requerido para as atividades de *overhead* que não são incluídas na WBS, mas que são claramente necessárias (JALOTE, 1999). Além disso, esse tipo de abordagem exige um esforço considerável e um custo com relação ao detalhamento de todas as possíveis atividades do projeto.

Segundo Boehm (1981), a abordagem *bottom-up* é complementar à abordagem *top-down*. A precisão das estimativas também depende de um detalhamento maior das informações do projeto e de uma base histórica de projetos semelhantes já concluídos.

As abordagens de estimativas descritas podem utilizar dados já documentados de projetos anteriores similares, para fornecer suporte à elaboração de novas estimativas de projetos. Para isto, os seguintes fatores devem ser considerados, por influenciarem a estimativa do esforço do projeto: o mesmo ciclo de vida do projeto, a mesma metodologia de desenvolvimento, as mesmas ferramentas e o uso de uma equipe de projeto com habilidades e experiências similares.

3.2 O Processo de Estimar

O propósito principal de um processo de estimativa é prestar informações que beneficiem o planejamento e o controle dos projetos de software o mais cedo possível durante seu ciclo de vida (VAZQUEZ *et al*., 2007).

Segundo Sommerville (2007), estimar envolve responder as questões seguintes:

- Qual o esforço requerido para realizar a atividade?
- Qual o tempo necessário para completar cada atividade?
- Qual o custo total de cada atividade?

Ao final do projeto devem-se registrar as medidas reais de esforço, prazo e custo construindo assim uma base histórica da organização. Neste contexto, apresentam-se dois processos para estimar um projeto de software: o primeiro é proposto por Vazquez *et al.* (2007), segundo o qual, um processo de estimativa de um projeto de software envolve quatro atividades básicas (Figura 3.1):

- Estimar o tamanho do produto a ser desenvolvido

- Estimar o esforço empregado na execução do projeto
- Estimar o prazo (duração) do projeto
- Estimar o custo do projeto

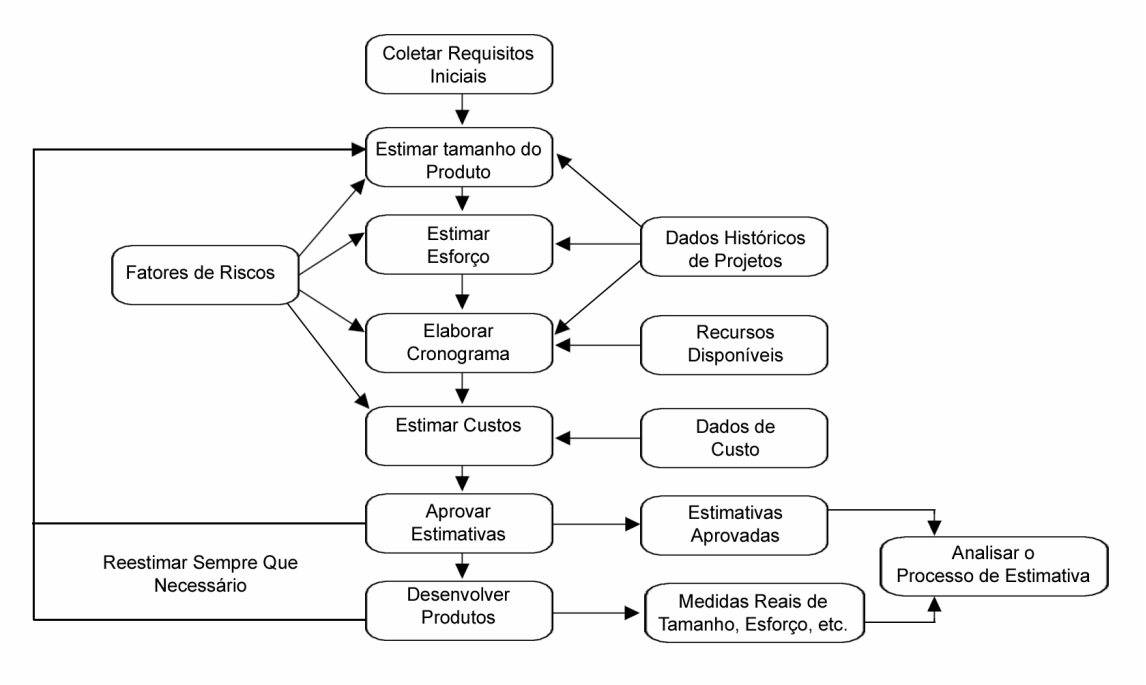


Figura 3.1 – Processo de estimativa de um projeto de software (VAZQUEZ *et al.*, 2007)

O segundo processo, proposto por AGARWAL *et al.* (2001), coloca as estimativas de tamanho apoiando as de esforço que por sua vez, apóiam as de custos e prazos (Figura 3.2). Segundo o autor, o processo de obtenção dos prazos e custos estimados para um projeto de software depende do seu ciclo de vida.

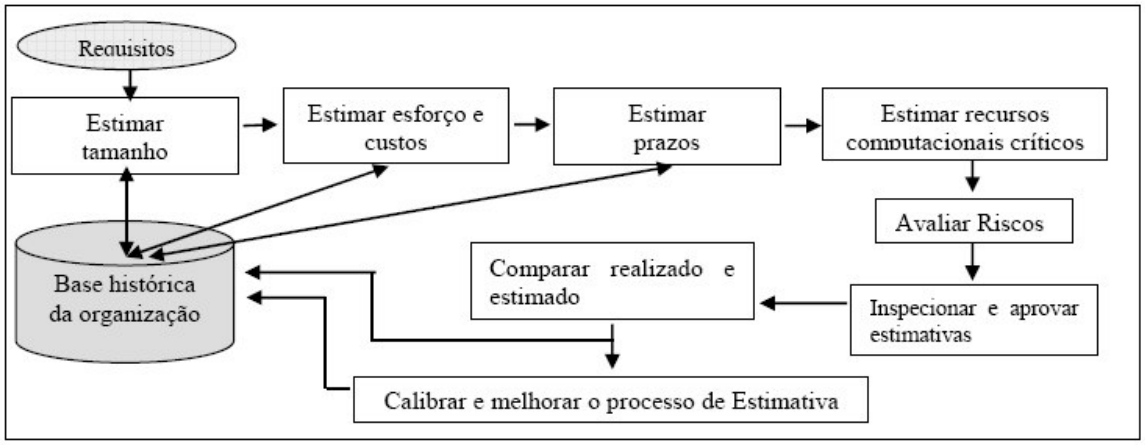


Figura 3.2 – Processo de estimativas segundo Agarwal – (AGARWAL *et al.*, 2001)

Apesar de existirem processos conhecidos para se estimar um projeto, em muitas organizações ou não existe o processo ou não se consegue a precisão necessária. Lederer e Prasad (1992) *apud* Pfleeger (2007) investigaram as práticas utilizadas para a estimativa de custos de 115 empresas. Trinta e cinco por cento dos gerentes pesquisados, em uma escala de Likert de cinco pontos, indicaram que suas estimativas eram "moderadamente insatisfatórias" ou "muito insatisfatórias". As principais causas identificadas pelos gerentes pesquisadores incluíam:

- Freqüentes solicitações de mudanças, feitas pelo usuário
- Tarefas negligenciadas
- Falta de entendimento dos usuários com relação a suas próprias exigências
- Análise insuficiente no desenvolvimento de uma estimativa
- Falta de coordenação do desenvolvimento do sistema, dos serviços técnicos, das operações, do gerenciamento de dados e de outras funções, durante o desenvolvimento
- Falta de um método adequado ou de diretrizes para a estimativa

Se por um lado é muito importante conhecer as estimativas no início do projeto, por outro lado, é consenso entre autores, que a estimativa inicial de um produto de software sempre terá um percentual de distorção por conta de fatores tais como: o não conhecimento total das necessidades do produto, a falta de detalhamento ou até mesmo o acréscimo de escopo.

Durante todo o ciclo de vida do software as estimativas devem ser refeitas, pois as estimativas tornam-se mais precisas à medida que se conhece mais sobre o projeto (PFLEEGER, 2007). A Figura 3.3 ilustra como a incerteza no começo do projeto pode afetar a precisão das estimativas do custo e do tamanho do projeto (BOEHM *et al.,* 1995). As estrelas representam o tamanho estimado de projetos reais e os sinais de adição referem-se às estimativas de custo. Note-se que quando as especificidades do projeto ainda não são conhecidas, a estimativa pode diferir do custo real final em até quatro vezes (PFLEEGER, 2007).

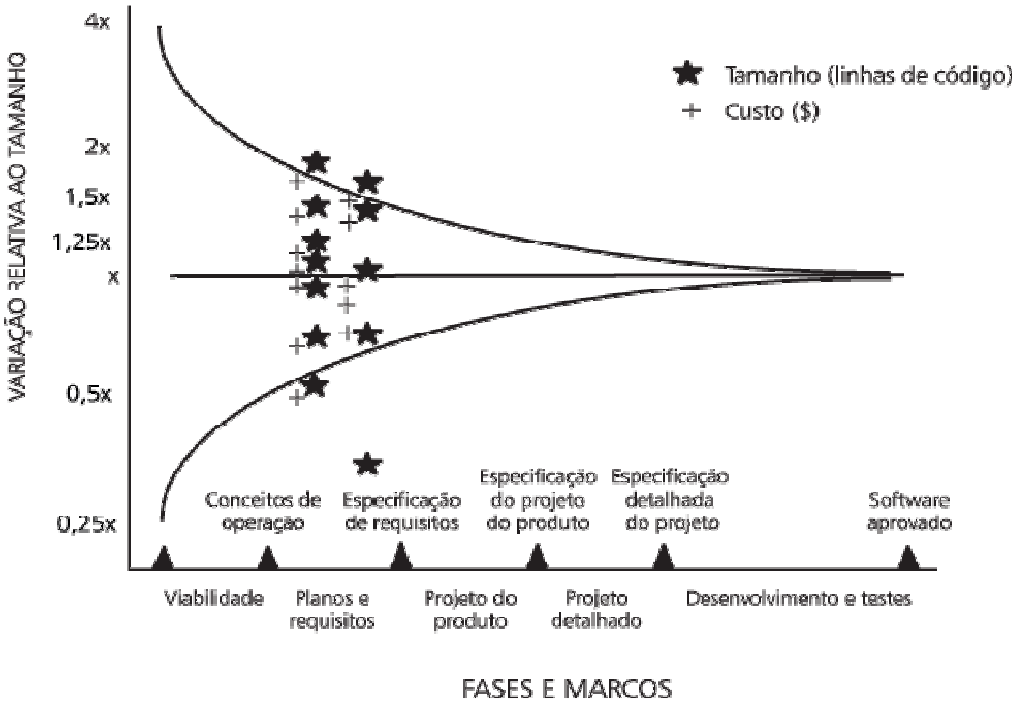


Figura 3.3 – Mudanças na precisão das estimativas, à medida que o projeto progride (BOEHM *et al*., 1995)

O PMBOK (2004) também menciona o fato, quando estabelece uma margem de erro aceitável nas estimativas realizadas a cada etapa do ciclo de vida. As estimativas de custos podem se beneficiar do refinamento durante o andamento do projeto para refletir os detalhes adicionais disponíveis. A exatidão de uma estimativa de projeto irá aumentar conforme o projeto se desenvolve por todo o seu ciclo de vida. Por exemplo, um projeto na fase de iniciação poderia ter uma variação em sua estimativa na faixa de -50 a +100%. Numa etapa posterior do projeto, conforme mais informações são conhecidas, a variação de suas estimativas poderia se reduzir a uma faixa de -10 a +15%. Em algumas áreas de aplicação, existem diretrizes para definir quando esses refinamentos são realizados e qual o grau de exatidão esperado (PMBOK, 2004).

3.2.1 Estimativa de Tamanho

Determinar o tamanho de um projeto de software é uma das primeiras e principais atividades relacionadas às estimativas a serem efetuadas durante o ciclo de vida do projeto. Há autores que destacam que a estimativa de tamanho é uma das atividades mais difíceis de

serem realizadas em uma organização de desenvolvimento de software, principalmente quando é realizada no início do projeto (ROSS, S.d).

O tamanho do software significa a quantidade de trabalho a ser executado no desenvolvimento de um projeto em uma unidade de medida especificada. Cada projeto pode ser estimado de acordo com seu tamanho físico (que pode ser medido por meio da especificação de requisitos, análise, projeto e código), com base nas funções que o usuário obtém, na complexidade do problema que o software irá resolver e no reuso do projeto (que mede o quanto o produto será copiado ou modificado a partir de um outro produto existente) (FENTON; PFLEEGER, 1997; McPHEE, 1999).

Segundo McPhee (1999), a estimativa de tamanho de software é um processo pelo qual uma pessoa ou um grupo de pessoas estima o tamanho de um produto de software. Para Ross (S.d), o tamanho geralmente tem impacto na solução técnica e na gestão do projeto, já que estimativas imprecisas podem levar ao fracasso do projeto. Neste contexto, a precisão das estimativas de tamanho torna-se fundamental para a elaboração de cronograma e orçamento realistas, pois essas estimativas constituem-se na base para a derivação das estimativas de esforço, prazo, e custo (SEI, 2002).

3.2.2 Estimativa de Esforço, Prazo e Custo

De forma genérica, esforço pode ser entendido como a quantidade de trabalho necessária para completar uma atividade ou outro elemento de um projeto e geralmente é expresso como um total de horas, dias, meses ou semanas gastos por um grupo de pessoas na realização de suas atividades (PMBOK, 2004). O PMBOK também ressalta que o esforço não deve ser confundido com duração, que pode ser definida como o tempo necessário para completar uma atividade ou outro elemento de um projeto (KOSLOSKI, 2005).

O esforço estimado para o projeto pode ser obtido a partir do cálculo da estimativa de tamanho do produto. O esforço total de um projeto é calculado com base em seu processo de desenvolvimento. Esse processo envolve muito mais do que a simples atividade de codificação do software. Elaboração de documentos, implementação de protótipos, projeto do produto a ser entregue, revisão e teste do código levam uma grande fatia de todo o esforço do projeto (PETERS, 1999) *apud* Monteiro (2005).

Para estimar o esforço e prazo, é preciso que seja selecionada uma abordagem para a obtenção da estimativa (modelos paramétricos, analogia, julgamento de especialistas, modelo de estimativas baseado em atividades, relações simples de estimativas).

O esforço de desenvolvimento ou manutenção de um sistema, ou seja, a quantidade de trabalho necessário para produzir ou manter um sistema demandará um determinado prazo de execução, dependendo da quantidade de recursos alocados ao projeto.

Para a maioria dos projetos, o maior componente do custo é o esforço. Deve-se determinar quantas pessoas/dias de esforço serão necessários para completar o projeto. O esforço é, certamente, o componente do custo com maior grau de incerteza (PFLEEGER, 2007) e a sua distribuição pelas tarefas a serem realizadas, juntamente com a alocação de pessoas da equipe, são importantes variáveis para a determinação dos prazos de execução dos projetos de desenvolvimento de software (PRESSMAN, 2000; BARCELLOS, 2003; PUTNAM e MAYERS, 2003).

A estimativa de prazo de maneira simplificada pode ser dada pela razão entre o esforço previsto (geralmente em número de horas trabalhadas) e a quantidade de recursos alocada na execução do projeto, conforme equação abaixo:

$$Prazo = \frac{\text{Esforço}}{\text{Quantidade de Recursos}}$$

Segundo McConnell (1996) e Peters (1999) *apud* Monteiro (2005), a estimativa de prazo pode ser obtida através da equação:

$$\textbf{\textit{Prazo}}\ \textbf{(meses)} = 3{,}0 * \textbf{Esforço (meses)}^{1/3}$$

Estimativas de prazo e custo devem ser elaboradas para todos os projetos de software, pois é objetivo básico na execução de um projeto realizá-lo dentro de prazos e custos estimados conforme o contrato. O grau de formalismo para se determinar essas estimativas pode diferir de um projeto para outro, dependendo de suas necessidades, de métodos prognósticos e da disponibilidade de ferramentas. Em projetos de software, o custo é comumente proporcional ao esforço despendido para sua construção, onde o trabalho humano é o principal recurso a ser consumido. Conseqüentemente, o custo é com freqüência associado a homens-mês ou homens-hora (MONTEIRO, 2005).

O custo total das horas trabalhadas pode ser obtido pelo produto da estimativa de esforço do projeto (em horas) e valor de uma hora trabalhada (R$ por hora). Um custo total mais preciso pode ser obtido por meio do valor das horas trabalhadas específicas de cada

recurso utilizado no projeto (recursos técnicos, recursos de suporte, gerente de projeto, etc.). Uma outra maneira de se obter o custo total das horas trabalhadas é determinar o percentual do esforço total do projeto a ser realizado em cada etapa do processo pelos recursos do projeto (MONTEIRO, 2005).

3.3 Métricas de Estimativa de Software

Em geral, as métricas de estimativas de tamanho de software são baseadas em modelos paramétricos. Desde 1960, avaliar o tamanho do software a ser construído tem sido uma preocupação no contexto do desenvolvimento de software, o que levou a definição de diferentes abordagens.

Diversos aspectos do projeto foram observados como influências importantes na estimativa (PFLEEGER, 2007):

- Complexidade do sistema proposto
- Necessidade de integração com sistemas existentes
- Complexidade dos programas no sistema
- Tamanho do sistema expresso em número de funções ou programas
- Capacidade dos membros da equipe de projeto
- Experiência da equipe do projeto com a aplicação
- Freqüência ou extensão prevista de potenciais mudanças nos requisitos dos usuários
- Experiência da equipe do projeto com a linguagem de programação
- Sistema de gerenciamento de banco de dados
- Número de membros da equipe do projeto
- Quantidade de padrões de programação e documentação
- Disponibilidade de ferramentas tais como geradores de aplicação
- Experiência da equipe com o hardware

Para resolver a necessidade de produzir estimativas precisas, os engenheiros de software têm desenvolvido técnicas para compreender a relação entre o esforço e as características de pessoal, os requisitos do projeto e outros fatores que podem afetar o tempo, o esforço e o custo para se desenvolver um sistema de software (PFLEEGER, 2007). A seguir são apresentadas algumas formas utilizadas para calcular as estimativas de tamanho de software.

3.3.1 Linhas de Código - LOC (*lines of code*)

A técnica de mensuração por linhas de código é uma das mais antigas medidas de tamanho de projeto de desenvolvimento de software. Ela consiste na contagem da quantidade do número de linhas de código de um programa de software. Ela considera o software sob a perspectiva de sua estrutura interna e é aplicada nas fases finais do projeto (MISIC e TESIC, 1998).

Como o próprio nome sugere, propõe que o tamanho de um sistema deve ser medido pela contagem das linhas de seu código fonte. Como a quantidade exata de linhas de código só é conhecida após a conclusão do sistema, a estimativa pode ser feita de duas formas (SIMON, 2000):

- Utilizando-se variáveis de estimativa usadas para cada elemento do sistema.
- Com o uso de dados de projetos anteriores, juntamente com variáveis de estimativa, projetando-se o custo e esforço de novos projetos.

Como vantagens dessa técnica podem ser citadas: a possibilidade de automatizar medições e a facilidade de uso de dados históricos (ROSS, S.d); simplicidade e farta literatura disponível (PRESSMAN, 2000); serve como medida de normalização para qualidade de software, como, por exemplo, quantidade de defeitos por linhas de código (FENTON, 1999).

ROSS (S.d) cita como desvantagens das linhas de código, a dificuldade de comparar produtividades entre linguagens diferentes. Além dessa, percebe-se que a métrica LOC é muito frágil e imprecisa, tendo várias desvantagens:

- A definição de linha de código é obscura, sem padrões, não estando claro se deve incluir ou não declarações de dados, macro-instruções, etc. (SIMON, 2000).
- É uma medida técnica, sem significado para o usuário (DUMKE, 1999; ROSS, S.d).
- Não é consistente, pois algumas linhas de código são mais trabalhosas que outras (DEMARCO, 1989).
- Apresenta problemas de definição para linguagens não procedurais (PRESSMAN, 2000).

Além dessas desvantagens, Jones (1994) *apud* Fenton e Pfleeger (1997) ressalta que uma contagem em LOC depende do grau de reuso e da linguagem de programação e pode ser até cinco vezes superior a uma outra estimativa, devido às diferenças das técnicas de medição de linhas em branco, linhas de comentário, declaração de dados e linhas de instruções. Essa

técnica penaliza programas pequenos e bem projetados, não se adapta às linguagens não procedimentais e é de difícil obtenção na fase inicial de planejamento.

LOC foi bastante utilizada até meados da década de setenta. A partir daí, surgiram diversas linguagens de programação e, conseqüentemente, a necessidade de se ter outras formas de estimar o tamanho de software.

3.3.2 Análise de Pontos de Função – APF

A Análise de Pontos de Função (APF) ou *Function Point Analisys* (FPA) mede as funcionalidades fornecidas por um software do ponto de vista de seu usuário. Ponto de função é a sua unidade de medida, que tem por objetivo tornar a medição independente da tecnologia utilizada para a construção do software (IFPUG, 2000). Foi desenvolvida por Allan Albrecht em 1979, buscando mapear as questões pertinentes à estimativa e avaliação de produtividade no desenvolvimento de software em ambientes heterogêneos. Em 1986, um grupo de usuários da FPA formou o *International Function Point User Group* (IFPUG), o qual é responsável por manter seus associados informados a respeito das novas atualizações da técnica. A FPA é uma das primeiras técnicas a medir o tamanho do software com alguma precisão. Ela busca medir o que o software faz, e não como ele foi construído.

Segundo Pressman (2000), a FPA é orientada à função, derivada a partir de medidas diretas, que dimensionam o software considerando a funcionalidade entregue ao usuário final. Em síntese, o projeto lógico deve ser inteiramente decomposto em funções de acordo com os arquivos de dados, interfaces, entradas, saídas e consultas, atribuindo-se pesos a cada uma dessas funções. Esses pesos são multiplicados pelas quantidades de cada função e posteriormente somados. Finalmente, o somatório é ajustado conforme uma análise das características gerais de complexidade do sistema.

A FPA visa estabelecer uma medida de "tamanho" do software, em Pontos por Função (PF) (unidade de medida para software assim como a hora é unidade de medida para tempo), através da quantificação das funções implementadas sob o ponto de vista do usuário (LONGSTREET, 2002). É mais utilizada no final das fases de análise e projeto.

O processo de contagem da FPA é realizado em sete etapas:

i. **Determinar o tipo de contagem:**

- Contagem de Pontos de Função de projetos de desenvolvimento
  - Mede as funcionalidades fornecidas ao usuário na instalação inicial de um novo software

- Contagem de Pontos de Função de projetos de manutenção
  - Mede modificações que incluem, alteram ou excluem funcionalidades em um software existente
- Contagem de Pontos de Função de aplicações já existentes
  - Mede as funcionalidades de uma aplicação instalada

ii. **Identificar o escopo de contagem e a fronteira da aplicação:**

- Define quais as funcionalidades objeto de cada contagem
- Identifica os sistemas, aplicações ou seus componentes que serão dimensionados
- A fronteira é determinada baseada na visão do usuário
  - O foco é no que o usuário pode ver e descrever
  - Baseada na perspectiva do negócio e não na implementação tecnológica

iii. **Contar as funções de dados:**

- Arquivos lógicos internos (ALI) – grupo de dados ou informações de controle reconhecidos pelo usuário logicamente relacionados, cuja manutenção é efetuada dentro da fronteira da aplicação. A principal função de um ALI é armazenar dados mantidos por um ou mais processos elementares da aplicação sendo contada.
- Arquivos de interface externa (AIE) - grupo de dados ou informações de controle reconhecidas pelo usuário logicamente relacionado, referenciado pela aplicação, mas mantido dentro da fronteira de outra aplicação. O principal objetivo de um AIE é armazenar dados referenciados por um ou mais processos elementares da aplicação sendo contada.

iv. **Contar as funções transacionais:**

- Entradas externas (EE) – processam dados ou informações de controle vindos do lado de fora da fronteira da aplicação. A principal função de um EE é manter um ou mais arquivo lógico interno e/ou alterar o comportamento do sistema.
- Saídas externas (SE) – enviam dados ou informações de controle para fora da fronteira de aplicação. A principal intenção de uma SE é apresentar dados ao

usuário utilizando processamento lógico adicional a recuperação de dados ou informação de controle. A lógica de processamento deve conter fórmula matemática ou cálculo, criar dados derivados, manter um ou mais arquivo lógico interno e/ou alterar o comportamento do sistema.

- Consultas externas (CE) – resulta na recuperação de dados ou informações de controle enviados para fora da fronteira da aplicação. A principal função de uma CE é apresentar informação ao usuário por meio da recuperação de dados ou informações de controle de um ALI ou AIE. A lógica de processamento não deve conter fórmulas matemáticas ou cálculos, criar dados derivados, manter um ou mais ALI, nem alterar o comportamento do sistema.

v. **Determinar os pontos de função não ajustados:**

- Consiste em somar todos os pontos de função obtidos da contagem de funções de dados e transacionais (etapas *iii e iv*).

vi. **Determinar os valores dos fatores de ajustes**

- Consiste na pontuação de cada uma das 14 características gerais da aplicação (Tabela 3.2) segundo uma escala de 6 níveis de influência, variando entre 0 (nenhuma influência) a 5 (máxima influência). Após a pontuação individual, o fator de ajuste é calculado pela seguinte equação:

  Fator de Ajuste = (Σ níveis de influência x 0,01) + 0,65

  Estas características influenciarão a complexidade do software e consequentemente seu tamanho.

vii. **Calcular os pontos de função ajustados**

- Os pontos de função ajustados são calculados a partir dos pontos de função não ajustados determinados na etapa 'v' e do fator de ajuste determinado na etapa '*vi*', que ajusta o valor da contagem segundo a avaliação das características gerais da aplicação, de acordo com a seguinte fórmula:

PF_DESENVOLVIMENTO = PF_NAOAJUSTADO * FATOR _AJUSTE

Apesar de a FPA ser mais utilizada para estimativa de tamanho de software, ela tem apresentado limitações por ter sido desenvolvida com base nas técnicas estruturadas. Uma

limitação da FPA refere-se à tentativa de seu uso na medição do tamanho funcional de outros tipos de software como o de tempo real, científicos e orientados a objetos (JONES, 1994; WHITMIRE, 1992; REIFER, 1990; MUKHOPADHYAY; KEKRE, 1992, citados por ABRAN *et al.* (2000); RIBU, 2001).

Tabela 3.2 – Características gerais do sistema para o cálculo do fator de ajuste (KOSLOSKI, 2005), (IFPUG, 2000)

Característica	Descrição
Comunicação de dados	Dados ou informações de controle são enviados e recebidos pela aplicação por meio de facilidades de comunicação. Esta característica mede o grau com o qual a aplicação utiliza estas facilidades.
Dados distribuídos	Descreve o grau no qual a aplicação transfere dados entre seus componentes físicos.
Desempenho	Descreve o grau no qual o tempo de resposta e o desempenho esperado para a aplicação influenciam o seu desenvolvimento.
Configuração da máquina	Descreve o grau no qual as restrições computacionais influenciam o desenvolvimento da aplicação
Taxa de transação	Descreve o grau no qual a taxa (volume) de transações de negócios influencia o desenvolvimento da aplicação
Entrada de dados on-line	Descreve o grau no qual os dados são entrados ou recuperados por meio de transações interativas, pela aplicação.
Eficiência do usuário final	Esta característica geral descreve o grau de facilidades para o usuário da aplicação sendo mensurada.
Atualização on-line de dados	Descreve o grau no qual ALIs são atualizados de forma on-line
Complexidade de Processamento	Descreve o grau de influência na aplicação de processamentos lógicos complexos
Reusabilidade	Descreve o grau no qual a aplicação e seu código foram especialmente projetados, desenvolvidos e suportados para serem utilizados em outras aplicações.
Facilidade de instalação	Descreve o grau no qual conversões provenientes de outros ambientes influenciam o desenvolvimento da aplicação.
Facilidade de operação	Descreve o grau no qual a aplicação atende a aspectos operacionais tais como: processos de inicialização, cópias de segurança (backup) e recuperação a falhas.
Múltiplos locais	Descreve o grau no qual a aplicação foi desenvolvida para diferentes ambientes de hardware e software.
Facilidade de mudanças	Descreve o grau de facilidade de modificações da aplicação em seus processamentos lógicos ou estrutura de dados.

3.3.3 NESMA

A NESMA (*Netherlands Software Metrics Users Association*) é uma associação de usuários de métricas que tem proposto alternativas de contagem utilizando a APF, de forma a

possibilitar a mensuração de um produto de software no início do processo de desenvolvimento, mesmo não possuindo todas as informações (VAZQUEZ *et al.*, 2007).

A NESMA, criada em 1989, desenvolveu e mantém práticas próprias de contagem (NESMA, 1996), incorporando os mesmos conceitos, metodologia e regras da FPA, versão 4.1 (IFPUG, 2000). Além disso, a NESMA também elaborou guias práticos e formas simplificadas de contagens de pontos de função.

A NESMA reconhece três tipos de contagem de pontos de função, os quais apresentam-se resumidamente:

- Contagem de pontos de função estimada:
  - A única diferença em relação à contagem usual de pontos de função é que a complexidade funcional não é determinada individualmente para cada função, mas pré-definida para todas elas.
  - Determinam-se todas as funções de todos os tipos (ALI, AIE, EE, SE, CE)
  - Segue as regras do manual da APF, versão 4.1 (IFPUG, 2000), mas considera todas as funções de dados como sendo de complexidade baixa e todas as funções de transação como de complexidade média.
  - Calcula-se o total de pontos de função não ajustados
  - Segundo NESMA (2005) a margem de erro deste método é considerada baixa, quando comparado ao método detalhado de contagem de PF.
- Contagem de pontos de função indicativa:
  - Determina-se a quantidade das funções do tipo dado (ALI e AIE)
  - Calcula-se o total de pontos de função não ajustados da aplicação da seguinte forma: tamanho indicativo (pf) = 35 x número de ALIs + 15 x número de AIEs
  - Esta estimativa é baseada somente na quantidade de arquivos lógicos existentes (ALIs e AIEs). A contagem indicativa é baseada na premissa de que existem aproximadamente três EEs (para adicionar, alterar, e excluir dados do ALI), duas SEs, e uma CE na média para cada ALI, e aproximadamente uma SE e uma CE para cada AIE.
  - Neste método, são associados 35 PF para cada ALI e 15 PF para cada AIE identificados. O tamanho do software é a soma dos pontos de função resultantes destas associações.
- Contagem de pontos de função detalhada:
  - Determinam-se todas as funções de todos os tipos (ALI, AIE, EE, SE, CE)

- Determina-se a complexidade de cada função (Baixa, Média, Alta)
- Calcula-se o total de pontos de função não ajustados

As contagens indicativa e estimada foram idealizadas pelo NESMA para serem utilizadas em etapas iniciais do ciclo de vida de desenvolvimento do sistema, onde ainda não existem definições detalhadas dos requisitos da aplicação, isto é, em momentos bem cedo no processo de estimativa de tamanho.

3.3.4 COCOMO

Em 1981, Boehm (BOEHM, 1981) apresentou o COCOMO (*Constructive Cost Model*), que é uma hierarquia de modelos de estimativas de software. Essas estimativas compreendem esforço, prazo, custo e tamanho da equipe para o desenvolvimento de produtos de software. O método pode ser aplicado nas várias fases do ciclo de desenvolvimento, fornecendo resultados mais satisfatórios em estágios mais avançados do projeto. O COCOMO busca estimar esforço, prazo, custo e tamanho de equipe, necessários ao desenvolvimento do software, desde que se tenha como premissa, a dimensão do mesmo. Existem 3 modelos categorizados da seguinte forma:

- COCOMO Básico – versão aplicável a grande maioria de projetos de software. Calcula o esforço e o custo de desenvolvimento de software em função do tamanho do programa estimado em linhas de código.
- COCOMO Intermediário – calcula o esforço de desenvolvimento de software em função do tamanho do programa e de um conjunto de "direcionadores de custos", que contêm avaliações subjetivas do produto, do hardware, da equipe e dos atributos do projeto.
- COCOMO Detalhado – Reúne todas as características da versão intermediária com uma avaliação do impacto dos direcionadores de custo em cada fase (análise, projeto, etc.) do processo de engenharia de software.

O modelo COCOMO está definido para três classes de projetos:

- Modo orgânico: projetos de software relativamente pequenos e simples, nos quais pequenas equipes com boa experiência em aplicações trabalham em um conjunto de requisitos não muito rígidos.

- Modo semi-destacado: projetos de software intermediários em tamanho e em complexidade, nos quais equipes com níveis heterogêneos de experiência devem alcançar uma combinação de requisitos variando de rígidos a não muito rígidos.
- Modo embutido: projetos que devem ser desenvolvidos dentro de um conjunto rígido de restrições operacionais de hardware e software.

3.3.5 COCOMO II

O Modelo COCOMO II teve como precursor o COCOMO, também conhecido como COCOMO 81. Devido à idade dos projetos que embasaram o modelo, assim como sua incapacidade de lidar com ciclos de vida iterativos e com a utilização de componentes *Commercial-Off-The-Shelf* (COTS[1]), o COCOMO 81 é atualmente considerado obsoleto, tendo sido substituído por sua versão II, publicada em 2000.

O COCOMO II aplicado ao RUP – *Rational Unified Process* (2003) estima o esforço, prazo e equipe média para as fases de Elaboração e Construção. As fases de Iniciação e Transição são estimadas como percentuais da soma Elaboração+Construção. Os marcos que caracterizam o escopo abrangido pelo COCOMO II são:

LCO – *Lyfe Cycle Objectives* – Ponto no qual é escolhida uma possível arquitetura para o projeto (não necessariamente aquela que será de fato utilizada). Ocorre ao final da Iniciação.

IOC – *Initial Operational Capability* – Ponto no qual é concluído o desenvolvimento do software, estando o sistema pronto para entrega e teste final.

3.3.6 Pontos de Caso de Uso – UCP

A UCP (*Use Case Points*) é uma técnica de estimativa de tamanho de projeto de software orientado a objetos, criada por Karner (1993), onde foi desenvolvida a técnica de diagramação para o conceito de casos de uso. Posteriormente, Jacobson desenvolveu a "*Object-Oriented Software Engeneering* (OOSE)", metodologia centrada em casos de uso, técnica largamente utilizada na indústria para descrever e capturar os requisitos funcionais de software (RIBU, 2001).

Na UCP, Karner (1993) explora o modelo e a descrição de casos de uso, substitui algumas características técnicas propostas pela FPA, cria os fatores ambientais e propõe uma

[1] Produtos comerciais de prateleira.

estimativa de produtividade. Como esta técnica é centrada em casos de uso, sua eficácia depende de uma padronização de todos os casos de uso de uma instituição, pois um dos passos para a obtenção das métricas é a contagem de transações por caso de uso. Transações representam um conjunto de atividades atômicas, e são executadas totalmente ou não, sendo identificadas nos fluxos de eventos dos casos de uso (MONTEIRO, 2005).

A UCP tem contribuído para diminuir algumas dificuldades encontradas pelo mercado em relação à resistência de adoção de métodos de estimativas, porque é uma técnica simples, fácil de usar e rápida de aplicar, quando se têm as informações necessárias para realizar as estimativas (DAMODARAN; WASHINGTON, 2003).

O método leva em consideração os atores e os casos de uso do software e se baseia em diferentes pesos para a obtenção dos cálculos. É composta pelas etapas a seguir:

i. Classificação dos Atores

ii. Classificação dos Casos de Uso

iii. Cálculo dos pontos de casos de uso não ajustados

iv. Cálculo dos fatores técnicos

v. Cálculo dos fatores ambientais

vi. Cálculo dos pontos de casos de uso

vii. Cálculo do esforço

Para a obtenção da estimativa de esforço total do projeto, a UCP utiliza um fator de produtividade (PROD). O fator de produtividade pode ser calibrado de acordo com a produtividade da equipe de um projeto e, em geral, é dado em horas.

Karner (1993) propôs uma produtividade de 20 h.h (homens/hora) por ponto de caso de uso, para projetos com requisitos estáveis e com equipe experiente, e 28 h.h por ponto de caso de uso, para projetos com requisitos não estáveis e com equipe não experiente.

Anda *et al.* (2001) mostraram que este esforço pode variar entre 15 h.h por ponto de caso de uso para projetos com requisitos estáveis e com equipe experiente, e 30 h.h por ponto de caso de uso para projetos com requisitos não estáveis e com equipe não experiente.

A Tabela 3.3 apresenta um resumo das etapas seguidas pela UCP no cálculo das estimativas.

Tabela 3.3 - Resumo da métrica de estimativa UCP

Etapa	Regra	Saída
i	Classificação dos atores • Simples, P(peso) = 1 • Médio, P=2 • Complexo, P=3	*Unadjusted Actor Weight* UAW = Σ (Ator * P)
ii	Classificação dos casos de uso • Simples – até 3 transações, P=5 • Médio – de 4 a 7 transações, P=10 • Complexo – acima de 7 transações, P=15	*Unadjusted Use Case Weight* UUCW = Σ (*Use Case* * P)
iii	Cálculo dos pontos de casos de uso não ajustados (UUCP- *Unadjusted Use Case Point*)	*Unadjusted Use Case Point* UUCP = UAW + UUCW
iv	Cálculo dos fatores técnicos (TCF – *Thecnical Complexity Factor)* • Atribuir nota de 0 a 5 a cada fator • Cada fator tem um peso correspondente. Este peso pertence ao intervalo [-1, 2]. • TFactor corresponde ao somatório dos produtos entre o peso e a nota atribuída de cada um dos 13 fatores de complexidade técnica (Tabela 3.4).	*Thecnical Complexity Factor* $TCF = 0,6 + (0,01 * TFactor)$
v	Cálculo dos fatores ambientais (EF – *Environmental Factor*) • Atribuir nota de 0 a 5 a cada fator • Cada fator tem um peso correspondente. Este peso pertence ao intervalo [-1, 2]. • EFactor corresponde ao somatório dos produtos entre o peso e a nota atribuída de cada um dos 8 fatores ambientais (Tabela 3.5).	*Environmental Factor* $EF = 1,4 + (-0,03 * EFactor)$
vi	Cálculo dos pontos de casos de uso (UCP – *Use Case Points*) • Consiste no tamanho total do sistema	*Use Case Points* UCP = UUCP * TCF * EF
vii	Cálculo do esforço • Consiste no produto entre o tamanho e a produtividade.	Esforço = UCP * PROD

Os fatores de complexidade técnica utilizados na UCP são apresentados na Tabela 3.4.

Tabela 3.4 – Fatores de complexidade técnica (RIBU, 2001)

Fator	Descrição	Peso	Nota
T1	Sistemas Distribuídos	2.0	
T2	Desempenho da Aplicação	1.0	
T3	Eficiência do Usuário-Final (*on-line*)	1.0	
T4	Processamento Interno Complexo	1.0	
T5	Reusabilidade do código em outras aplicações	1.0	
T6	Facilidade de Instalação	0.5	
T7	Usabilidade (facilidade operacional)	0.5	
T8	Portabilidade	2.0	
T9	Facilidade de Manutenção	1.0	
T10	Concorrência	1.0	
T11	Características especiais de segurança	1.0	
T12	Acesso direto para terceiros	1.0	
T13	Facilidades especiais de treinamento	1.0	

Os fatores ambientais utilizados na UCP são apresentados na Tabela 3.5.

Tabela 3.5 – Fatores ambientais (RIBU, 2001)

Fator	Descrição	Peso	Nota
F1	Familiaridade com o processo de desenvolvimento	**1,5**	
F2	Experiência com a aplicação em desenvolvimento	**0,5**	
F3	Experiência com orientação a objeto	**1**	
F4	Capacidade de análise	**0,5**	
F5	Motivação	**1**	
F6	Requisitos estáveis	**2**	
F7	Colaboradores de meio período	**-1**	
F8	Dificuldade na linguagem de programação	**-1**	

3.3.7 Pontos de Caso de Uso Técnico – TUCP

A TUCP (Pontos de Caso de Uso Técnicos) é uma extensão da UCP (Pontos de Caso de Uso), proposta por Monteiro (2005), objetivando um cálculo mais acurado para as estimativas a partir de casos de uso.

O objetivo primordial da TUCP é conseguir uma maior precisão na contagem de pontos de caso de uso e fornecer uma visão mais detalhada das estimativas (tamanho, esforço, prazo e custos) por etapas do ciclo de vida do projeto, possibilitando um acompanhamento mais efetivo do projeto.

A TUCP congrega os pontos seguintes:

- Propor um Guia para a elaboração de casos de uso, influenciando fortemente no cálculo do tamanho dos casos de uso.
- Detalhar o conceito de transação no contexto de casos de uso, por afetar diretamente o cálculo dos pontos de casos de uso não ajustados.
- Refinar a contagem de pontos por casos de uso complexos, para evitar possíveis subestimativas, especialmente quando o número de transações é muito grande.
- Desatrelar os fatores técnicos de ambiente (EF) do cálculo do tamanho, por se entender que o tamanho é uma grandeza física que não deve ter seu valor alterado em função dos fatores técnicos de ambiente.
- Considerar apenas no cálculo do esforço os fatores ambientais (EFs), juntamente com o fator de produtividade.
- Detalhar o cálculo do esforço nas etapas do ciclo de vida.

- Granularizar o cálculo do tamanho do projeto por cada etapa do processo de desenvolvimento e por caso de uso. O somatório de todos esses tamanhos será o tamanho final para desenvolver o projeto, não incluindo os esforços indiretos.

O grau de precisão da TUCP depende diretamente de a organização possuir um padrão para a especificação de casos de uso, como também é dependente do entendimento claro do que seja o conceito de transação. Estas questões influenciam diretamente no cálculo do tamanho do projeto.

Outra questão que influencia fortemente o grau de precisão da técnica é a contagem das transações de cada caso de uso. A quantidade de transações em um caso de uso determina sua complexidade, podendo classificá-lo como simples, médio, complexo e *n*-complexo, conforme explicação adiante. A identificação das transações de um caso de uso não é trivial devido ao grau de subjetividade existente neste tipo de contagem. Assim, torna-se importante o entendimento claro do conceito de uma transação, no contexto de caso de uso:

Uma transação é cada passo dos fluxos de eventos (básico e ou alternativo) de um caso de uso, onde ocorra um evento entre um ator e o sistema e que deva ser executado por completo ou ainda a realização de algum processamento complexo nesses fluxos de eventos.

A TUCP realiza as atividades seguintes para calcular as estimativas do projeto:

- **Estimativa de Tamanho**

A estimava de tamanho da TUCP envolve as seguintes etapas:

i. Contagem dos atores (UAW)

ii. Contagem dos casos de uso (TUUCW)

iii. Cálculo dos pontos de casos de uso não ajustados (TUUCP)

iv. Cálculo dos fatores de complexidade técnica (TCF)

v. Cálculo dos pontos de caso de uso técnicos (TUCP)

- **Contagem dos Atores**

O peso total dos atores do sistema é calculado pela soma dos produtos dos atores de cada tipo pelo respectivo peso, UAW (*Unadjusted Actor Weight*), de forma semelhante à UCP (KARNER, 1993), segundo a Tabela 3.6:

Tabela 3.6 - Classificação dos atores

Tipo	Descrição	Peso
Simples	Aplicação com APIs definidas	1
Médio	Aplicação com interface baseada em protocolo ou interação de usuário baseado em linhas de comandos	2
Complexo	Interação de usuário através de interface gráfica ou página Web	3

- **Contagem dos Casos de Uso**

O peso total dos casos de uso do sistema é calculado a partir da Tabela 3.7, isto é, o TUUCW (*Technical Unadjusted Use Case Points Weight*).

Tabela 3.7 - Contagem dos casos de uso

Tipo	Descrição	Peso
simples	até 3 transações incluindo os passos alternativos	5
médio	de 4 a 7 transações incluindo os passos alternativos	10
complexo	de 8 a t transações incluindo os passos alternativos	15
n-complexo	acima de t transações	P_X

Os casos de uso com até *t* transações serão calculados da mesma maneira apresentada por Karner (1993). Acima de *t* transações, o tipo de caso de uso será denominado de *n*-complexo. O cálculo do peso dos casos de usos *n*-complexos é exibido nas Equações 3.1 e 3.2:

$$\text{TUUCW} = 15n + p \qquad (Eq.\ 3.1)$$

$$\text{n} = T / t \qquad (Eq.\ 3.2)$$

Nas Equações 3.1 e 3.2, T = número de transações do caso de uso, e p = o peso obtido, quando o resto (r) da divisão de T / t é aplicado ao peso original (*simples*, *médio*, e *complexo*) (Tabela 2). Assim sendo, se $r = 0$, $p = 0$; se $r \in [1, 3]$, $p = 5$; se $r \in [4, 7]$, $p = 10$; e se $r \in [8, (t - 1)]$, $p = 15$.

A definição do valor de *t* poderá depender das características da organização, ou até mesmo das características de um dado tipo de projeto. Utiliza-se $t = 11$ pelas razões apresentadas em Monteiro (2005). Por exemplo: um caso de uso com 13 transações ($T = 13$) e com $t = 11$, tem então, $n = 13 / 11$, e $r = 2$. Como $r \in [1, 3]$, então $p = 5$. Assim: TUUCW = 15 * 1+ 5 = 20

- **Cálculo dos Pontos de Casos de Uso não Ajustados**

O cálculo dos pontos de caso de uso técnicos não ajustados, TUUCP (*Technical Unadjusted Use Case Points*), é efetuado pela Equação 3.3:

$$\text{TUUCP} = \sum \text{UAW} + \sum \text{TUUCW} \qquad (Eq.\ 3.3)$$

- **Cálculo dos Fatores de Complexidade Técnica**

Os fatores de complexidade técnica, TCF (*Technical Complexity Factor*), apresentados na Tabela 3.4, são calculados de forma similar à UCP (KARNER, 1993).

Deve-se atribuir nota de 0 a 5 para cada fator. O TCF é obtido com a utilização da Equação 3.4, em que TFactor corresponde ao somatório dos produtos entre o peso e a nota atribuída de cada um dos 13 fatores de complexidade técnica da Tabela 3.4.

$$TCF = 0{,}6 + (0{,}01 * TFactor) \qquad (Eq.\ 3.4)$$

- **Cálculo dos Pontos de Caso de Uso Técnicos**

O cálculo dos pontos de caso de uso técnico, TUCP (*Technical Use Case Points*) ajustada é dado pela Equação 3.5:

$$TUCP = TUUCP * TCF \qquad (Eq.\ 3.5)$$

- **Estimativa de Esforço**

A estimava do esforço do projeto por meio da utilização da TUCP envolve as seguintes etapas: Cálculo dos fatores ambientais (EF), Cálculo da produtividade (PROD), Cálculo do esforço técnico (Esforço).

- **Cálculo dos Fatores Ambientais**

Os fatores ambientais, EF (*Environmental Factor*), apresentados na Tabela 3.5, são calculados de forma similar à UCP (KARNER, 1993). Da mesma maneira que a UCP, os fatores ambientais dizem respeito aos requisitos não-funcionais associados ao projeto, tais como experiência da equipe, estabilidade do projeto e motivação dos programadores.

Deve-se atribuir nota de 0 a 5 para cada fator. O fator ambiental (EF) é calculado pela Equação 3.6, onde o EFator é o somatório dos produtos entre o peso e a nota atribuída de cada um dos 8 fatores ambientais da Tabela 3.5.

$$EF = 1{,}4 + (-0{,}03 * EFator) \qquad (Eq.\ 3.6)$$

- **Cálculo da Produtividade**

O fator de produtividade pode ser calibrado de acordo com a produtividade da equipe por tipos de atividade de um projeto. Esse fator pode ser obtido baseado em uma base histórica organizacional, onde podem ser armazenados dados, como, por

exemplo, o fator de produtividade para cada etapa do ciclo de vida e características de projeto.

A TUCP utiliza a faixa de valor para produtividade proposta por Anda *et al*. (2001) que varia de 15 h.h por UCP para projetos onde a equipe seja considerada estável e experiente, e 30 h.h por UCP para projetos em que os requisitos não sejam estáveis e com uma equipe inexperiente.

- **Cálculo do Esforço Técnico**

Para obter a estimativa de esforço total para o projeto, a TUCP utiliza o fator de produtividade (PROD), o fator de ambiente (EF) e o tamanho TUCP, segundo a Equação 3.7. Neste caso, o fator de ambiente (EF) retorna ao cálculo do esforço total do projeto de forma semelhante à UCP.

$$\textbf{Esforço} = \textbf{TUCP} * \textbf{EF} * \textbf{PROD} \qquad (Eq.\ 3.7)$$

- **Estimativa de Esforço por Etapa do Ciclo de Vida**

Na UCP (KARNER, 1993), o fator produtividade é único para todo o projeto. No entanto, constatou-se que o fator de produtividade pode variar dependendo da etapa do ciclo de vida do projeto. Isto porque a experiência da equipe de desenvolvimento pode variar de uma etapa para outra, já que, normalmente, as pessoas envolvidas em uma etapa não são as mesmas de uma outra etapa. Além disso, outros fatores podem influenciar na produtividade tais como: tipo de equipe, linguagem, experiência, utilização de *frameworks*, etc.

A calibragem para a produtividade para cada etapa do ciclo de vida deve ser obtida da base histórica organizacional. Assim, o esforço por etapa do ciclo de vida do projeto é calculado na Equação 3.8:

$$\textbf{Esforço}_{\textbf{(Etapa)}} = \textbf{TUCP}_{\textbf{(Etapa)}} * \textbf{EF} * \textbf{PROD}_{\textbf{(Etapa)}} \qquad (Eq.\ 3.8)$$

Em decorrência da Equação 3.8, o esforço por etapa do ciclo de vida para o caso de uso ($\text{Esforço}_{(UC\_Etapa)}$) e esforço do caso de uso ($\text{Esforço}_{(UC)}$) podem ser obtidos conforme Equação 3.9:

$$\textbf{Esforço}_{\textbf{(UC\_Etapa)}} = \textbf{TUCP}_{\textbf{(UC\_Etapa)}} * \textbf{EF} * \textbf{PROD}_{\textbf{(Etapa)}} \qquad (Eq.\ 3.9)$$

- **Estimativa de Prazo e Custo**

As estimativas de prazo e custos podem ser obtidas a partir do tamanho estimado em TUCP, disponibilidade de recursos, restrições do projeto, entre outras questões.

Pode-se determinar o total do tempo (na unidade de tempo considerada), que será necessário ao projeto, multiplicando-se o valor da TUCP pelo tempo médio gasto para cada TUCP, segundo a Equação 3.10. Em geral, a unidade de tempo utilizada é hora. O valor do tempo médio em horas por TUCP é obtido historicamente na organização.

$$\mathbf{Prazo_{(unidade\_tempo)} = TUCP * Tempo\_médio_{(TUCP)}} \qquad (Eq.\ 3.10)$$

O uso de técnicas do tipo PERT (*Programme Evaluation Review Technique*) e CPM (*Critical Path Method*) pode auxiliar na determinação do prazo da finalização do projeto, levando em consideração caminhos críticos, alocação de recursos, seqüenciamento de atividades e dependência de tarefas.

A estimava de custo pode ser determinada utilizando-se as equações Equação 3.11 ou Equação 3.12. O custo é calculado na unidade monetária considerada (por exemplo, em Reais).

$$\mathbf{Custo_{(unidade\_monetária)} = TUCP * Valor_{(TUCP)}} \qquad (Eq.\ 3.11)$$

$$\mathbf{Custo_{(unidade\_monetária)} = Esforço * Valor_{(unidade\_esforço)}} \qquad (Eq.\ 3.12)$$

Esta maneira de calcular o custo está relacionada apenas com o custo de esforço relacionado aos casos de usos do projeto relacionados. Os gastos relacionados à gestão de projeto, gestão de configuração, atividades de qualidade (*Quality Assurance*) e despesas indiretas do projeto como aquisição ou aluguel de hardware e software, viagens com reuniões e testes no cliente, contas telefônicas (por exemplo: ligações de longas distâncias, vídeo-conferência, linhas dedicadas para testes, etc.), treinamentos, despesas com os postos de trabalho, entre outros custos não estão sendo considerados.

3.4 Estimativas de Manutenção de Software

De acordo com Parthasarathy (2007), em um cenário típico de solicitação de um serviço de manutenção, as atividades executadas desde a solicitação do serviço até sua entrega são apresentadas na Figura 3.4. Uma vez recebida a solicitação a equipe de manutenção executa os seguintes atividades:

i. Avaliar a solicitação, analisando seus impactos. A análise de impacto prevê os programas, telas, arquivos e outras partes do aplicativo que serão afetados com a manutenção. Segundo Pfleeger (2007), a análise de impacto é a avaliação dos muitos riscos associados com a alteração a ser realizada, incluindo as estimativas do efeito nos recursos, esforço, e cronograma. Pode-se utilizar a análise de impacto para ajudar a controlar os custos de manutenção

ii. Estimar o esforço e o tempo necessário para executar as mudanças, baseado nas informações obtidas por meio da análise de impacto

iii. Obter o aceite formal do usuário para executar as mudanças

iv. Modificar os programas, telas, arquivos e relatórios impactados com a mudança

v. Testar as mudanças realizadas. Uma vez finalizados os testes, o aplicativo alterado deve ser levado para produção

vi. Fechar formalmente a solicitação de mudança

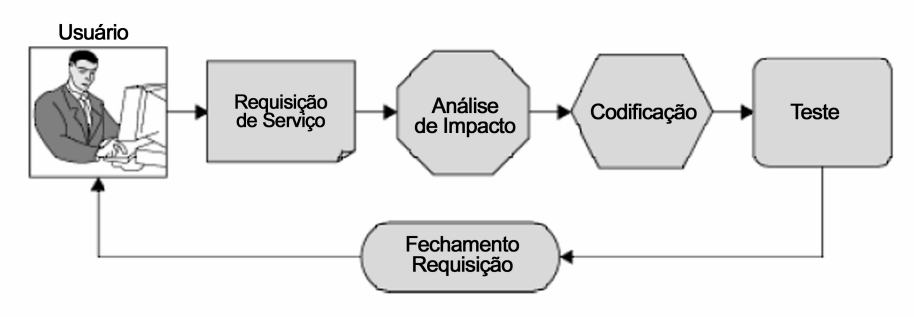


Figura 3.4 – Requisição de Serviço (PARTHASARATHY, 2007)

Conforme mencionado por diversos autores (APRIL, 1995; BOURQUE *et al.,* 1996; WEBSTER *et al*., 2004) as organizações consomem um tempo considerável no esforço de manutenção de seus aplicativos. Bem maior que o tempo dedicado ao desenvolvimento de novos sistemas. Isto implica que também o custo de manutenção desses aplicativos é muito superior ao custo do desenvolvimento de novos sistemas. Assim sendo, existe uma grande necessidade por técnicas de estimativas para projetos de manutenção de software, para que essas empresas possam gerenciar adequadamente esse tipo de projeto de software.

KUSTERS *et al.* (2001) relatam os seguintes resultados obtidos em uma pesquisa realizada junto a seis grandes organizações alemãs com a finalidade de dimensionar as dificuldades presentes na manutenção de software:

(i) Existe uma generalizada falta de percepção sobre o tamanho das manutenções de software, bem como do custo correspondente

(ii) O gasto é alto e os orçamentos têm sido extrapolados

(iii) É uma atividade de difícil planejamento

(iv) Somente uma, entre as seis organizações, registrava dados sobre seus processos de manutenção e os usava para planejamento

Koskinen (2003) faz considerações relacionadas a diversas técnicas utilizadas para estimar o custo de projetos de manutenção, mencionando que a maioria dos modelos usados para estimar os custos de projetos de software não leva em consideração as peculiaridades dos projetos de manutenção. Neste contexto, muitas organizações utilizam técnicas de estimativas conhecidas em seu processo de desenvolvimento, mas não a utilizam quando se trata de manutenções (FREIRE & BELCHIOR, 2007a).

3.4.1 APF para Projetos de Manutenção de Software

A Análise de Pontos de Função (APF) propõe medir apenas as manutenções que alteram os requisitos funcionais, identificados como "projetos de melhoria". Segundo o Manual de Práticas de Contagem do IFPUG (2000), uma contagem de um projeto de melhoria mede as alterações realizadas em uma aplicação existente com a finalidade de incluir, excluir ou modificar funcionalidades entregues. Não estão contempladas manutenções corretivas e preventivas.

Além disso, como a contagem é realizada no projeto como um todo e não somente no que foi efetivamente mantido, isso acaba por elevar o cálculo realizado. Segundo Calazans (2005), em 88% dos projetos da amostra realizada em seu estudo, o cálculo do esforço obtido com a utilização da APF ficou acima do tempo real gasto, para a efetiva realização da manutenção. As equipes que participaram do estudo reconheceram que, com relação aos projetos de melhoria, a APF superestima o tamanho.

Tran-Cao e Levesque (2002) demonstraram que a APF, quando aplicada a alguns projetos de manutenção, não apresentou bons resultados com relação a estimativas de tempo de execução quando comparadas com o tempo real.

3.4.2 UCP e TUCP para Projetos de Manutenção de Software

Percebe-se que a UCP (Pontos de Casos de Uso) bem como a TUCP (Pontos de Caso de Uso Técnico) são mais aplicáveis a projetos de desenvolvimento de software, pelos motivos seguintes:

- Não está previsto nessas técnicas uma etapa de avaliação de impacto para que sejam identificados os casos de uso que serão afetados com a manutenção, delimitando assim a quantidade dos casos de uso de um projeto de manutenção.
- Consideram em cada caso de uso todas as suas transações, independente se essas transações sofreram influência da manutenção solicitada ou não.
- Não leva em consideração que dependendo do tipo de manutenção (perfectiva, adaptativa, corretiva, preventiva) etapas do ciclo de vida podem não requerer esforço, ou requerer parcialmente.

Tal comportamento afeta o cálculo da estimativa de tamanho e consequentemente o cálculo da estimativa de esforço, prazo e custo do projeto.

3.5 Conclusão

Um dos aspectos cruciais do planejamento e gerenciamento de projetos é a compreensão de quanto o projeto provavelmente custará (PFLEEGER, 2007). E para se chegar à estimativa desse custo é necessário que anteriormente sejam calculadas as estimativas de tamanho, esforço e prazo desse projeto. Tais estimativas devem ser feitas o quanto antes, durante o ciclo de vida do projeto, pois elas afetam a distribuição de recursos e a viabilidade do projeto (PFLEEGER, 2007).

Apesar dessa necessidade, quanto mais cedo estas estimativas são realizadas, maior a dificuldade na obtenção de estimativas acuradas. Por esta razão, elas devem ser realizadas repetidamente durante o ciclo de vida do projeto. À medida que os aspectos do projeto se modificam a estimativa pode ser aprimorada, com base em informações mais completas sobre as características do projeto (PFLEEGER, 2007).

Muitas técnicas foram desenvolvidas objetivando permitir a determinação das estimativas dos projetos nas organizações. Técnicas propostas ao longo dos anos, que auxiliam a melhorar a gerência e planejamento dos projetos de software, na elaboração de propostas de negócios entre cliente e fornecedor, no gerenciamento de contratos de terceirização, possibilitando ainda avaliar a produtividade das equipes envolvidas e propor melhorias ao processo utilizado para desenvolvimento e manutenção de software.

Observa-se, porém, que apesar de existir a preocupação em aprimorar o cálculo das estimativas para projetos de manutenção de software, a maioria dos processos e técnicas propostos é mais aplicável a projetos de desenvolvimento ou a determinados tipos de manutenção. Revela-se de grande importância, estudar e adequar essas técnicas, para que se

considerem as características específicas de projetos dessa natureza. Some-se a isso, a necessidade de que a técnica proposta seja simples, fácil de usar, rápida de aplicar e que permita calcular as estimativas já nas etapas iniciais do projeto. É necessário reconhecer este cenário, para que as organizações possam identificar caminhos alternativos para a mensuração de projetos de manutenção.

A seguir, será apresentada a técnica TUCP-M (Pontos de Caso de Uso Técnico para Manutenção de Software) que objetiva estimar projetos de manutenção de software.

Capítulo 4

TUCP-M – Pontos de Caso de Uso Técnico para Manutenção de Software

Este capítulo propõe uma técnica para estimativas de manutenção de software, a TUCP-M (Pontos de Caso de Uso Técnico para Manutenção de Software), desenvolvida a partir da abordagem de casos de uso, estendendo a técnica TUCP (Pontos de Caso de Uso Técnico), proposta para estimativas de projetos de desenvolvimento de sistemas.

As estimativas dos projetos de software são apontadas como muito relevantes para o planejamento e gerenciamento desses projetos, conforme comentado no Capítulo 3, porque subsidiam a gerência das organizações em suas tomadas de decisão, nas negociações de novos contratos, avaliação de seus processos e da produtividade da equipe.

O planejamento construído para o desenvolvimento de um software estará comprometido, caso seja baseado em estimativas inapropriadas ou apenas baseado em experiências anteriores. Estimativas pouco acuradas podem levar ao fracasso todo um planejamento e estimativas baseadas apenas em experiências anteriores tornam-se pouco confiáveis quando a equipe é diferente da anterior ou quando o projeto for construído de forma terceirizada. Por mais capaz que seja a equipe técnica envolvida no projeto, a dificuldade de se estabelecer o tamanho de um projeto de software com base apenas na experiência passada é muito grande, pelo fato da dificuldade de se estabelecer as similaridades e as diferenças entre as funcionalidades dos softwares.

O cenário tende a se agravar para os casos em que o projeto a ser iniciado refere-se à manutenção de software. A existência de técnicas que possam estimar com simplicidade e precisão o tamanho, esforço, prazo e custo dos projetos relacionados à manutenção de software, surge como de vital importância no cenário das empresas que desenvolvem, consomem ou constroem seus próprios aplicativos. Entretanto, a maioria das técnicas existentes para estimativas de projetos é mais direcionada a projetos de desenvolvimento de software, havendo ainda lacunas a serem preenchidas no que se refere às estimativas de manutenção de software (FREIRE & BELCHIOR, 2007b).

Esse trabalho propõe uma técnica cujo objetivo é calcular estimativas de tamanho, esforço, prazo e custo de projetos de manutenção de software, desenvolvida a partir da abordagem de casos de uso, conforme mostrado nos itens que se seguem.

4.1 Objetivos e Pré-requisitos

A proposta de estimativas de manutenção de software a partir de casos de uso está baseada na TUCP (Pontos de Caso de Uso Técnicos) (MONTEIRO, 2005), que originalmente foi concebida para tratar estimativas de caso de uso para projetos de desenvolvimento de software. A esta proposta, chamaremos de **TUCP-M (Pontos de Caso de uso Técnico para Manutenção de Software)**.

A utilização da TUCP-M objetiva um cálculo mais acurado das estimativas de projetos de manutenção de software a partir de casos de uso, possibilitando ainda sua aplicação nas etapas iniciais do projeto. Como forma de aprimoramento constante da técnica, propõe a avaliação das razões que levaram à diferença entre os valores previsto e realizado.

De forma similar à TUCP, proporciona uma visão mais detalhada das estimativas para as principais etapas do ciclo de vida do software, viabilizando um acompanhamento mais efetivo do projeto. Também tem seu grau de precisão diretamente dependente de a organização possuir um modelo que seja utilizado para a especificação de casos de uso, como também é dependente do entendimento claro do que seja o conceito de transação. Estas questões influenciam diretamente no cálculo do tamanho do projeto, conforme já explicitado no Capítulo 3, na seção que trata da TUCP.

A TUCP-M é uma métrica de estimativa direta, dentro da categoria de modelos paramétricos e com abordagem *top-down*.

4.2 Características Gerais

A TUCP-M propõe, em linhas gerais, estimar determinado projeto de manutenção de software e avaliar os resultados alcançados, percorrendo as etapas apresentadas na Figura 4.1.

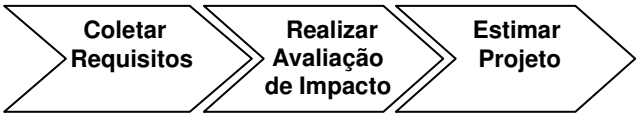


Figura 4.1 – TUCP-M – Estimar projeto

Após a execução do projeto devem-se coletar os valores reais referentes ao tamanho, esforço, prazo e custo e comparar com os valores estimados, objetivando melhorar continuamente a técnica (Figura 4.2).

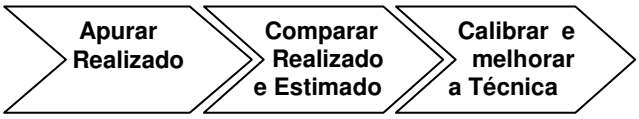


Figura 4.2 – TUCP-M – Avaliar resultados alcançados

4.2.1 Coletar Requisitos

O que dispara o início de um projeto de manutenção é a solicitação do usuário ou cliente, ou a identificação da necessidade de mudança, evolução, adaptação ou correção do software existente. Para isso, um documento de solicitação de mudança pode ser utilizado.

Nesse documento, devem ser identificadas e detalhadas as mudanças solicitadas. A aplicação da TUCP-M independe da utilização desse documento, entretanto, caso seja possível sua utilização, facilitará a identificação dos requisitos envolvidos na manutenção e conseqüentemente quais casos de uso serão mantidos. Nesse trabalho é proposto um documento de solicitação de mudanças (Apêndice A), que poderá servir como entrada para o pedido de mudança no software.

4.2.2 Realizar Avaliação de Impacto

- **Identificar requisitos**

Quando uma manutenção de software é solicitada, o primeiro entendimento que se deve ter é: quais requisitos novos estão sendo solicitados, quais deverão sofrer mudanças e quais deverão ser excluídos (FREIRE & BELCHIOR, 2007c). Esses requisitos irão sensibilizar um ou mais casos de uso já existentes, ou indicar a necessidade de desenvolvimento de um novo caso de uso, ou exclusão de um outro. É fundamental a realização de uma análise identificando quais requisitos foram incluídos, quais foram excluídos e quais foram alterados, pois isto servirá de base para o cálculo do tamanho do projeto de manutenção.

A identificação dos requisitos pode ser realizada com base no documento de solicitação de mudanças. Entretanto, ele não é obrigatório para a aplicação da TUCP-M. Existem casos em que a organização possui um sistema de gerenciamento de solicitações, cuja descrição das mudanças é apresentada em um campo texto livre. Pode-se ainda identificar os requisitos envolvidos na manutenção nas entrevistas de levantamento realizadas com os usuários gestores ou clientes.

Identificados os requisitos, o próximo passo é identificar os casos de uso afetados com a mudança e seus respectivos atores.

- **Identificar os casos de uso afetados e seus respectivos atores**

A identificação dos casos de uso afetados pode ser feita, por exemplo, por meio de uma matriz de rastreabilidade, onde cada um dos requisitos do sistema estará relacionado a um ou mais caso de uso. Dessa forma, cada requisito identificado no projeto de manutenção apontará os casos de uso afetados.

Para cada caso de uso identificado, devem-se verificar os atores envolvidos e os passos do fluxo de eventos afetados em cada caso de uso identificado.

- **Identificar passos do fluxo de eventos**

Identificar, para cada caso de uso afetado, os passos do fluxo de eventos que precisarão ser mantidos no documento de Especificação de Caso de Uso. Recomenda-se o *template* desse documento proposto por Monteiro (2005). Esses passos podem ser do tipo: *incluído*, *alterado* ou *excluído*:

- *Passos incluídos*: são os novos passos inseridos, que possam ser contados como transação.
- *Passos alterados:* são os passos que sofreram alguma modificação em sua estrutura, isto é, foram afetados pela manutenção, e que possam ser contados como transação.
- *Passos excluídos:* são os passos que tiveram sua descrição retirada do fluxo de evento, e que eram contados como transação.

4.2.3 Estimar Projeto

- **Efetuar a contagem das transações**

Efetuar a contagem apenas das transações dos passos dos fluxos de eventos dos casos de uso que foram mantidos, isto é, dos passos que foram incluídos, alterados ou excluídos.

No contexto desse trabalho, uma transação é cada passo dos fluxos de eventos (básico ou alternativo) de uma especificação de caso de uso, em que ocorra um evento entre um ator e

o sistema e que este deva ser executado por completo, ou ainda a realização de algum processamento complexo.

Um dos problemas detectados para a realização da manutenção de software é a inexistência ou falta de atualização da documentação dos sistemas legados, o que poderia induzir alguém a concluir que a utilização da TUCP-M estaria inviabilizada. Ao contrário, nos casos em que o sistema a ser mantido não possuir a especificação dos casos de uso, é possível realizar a contagem das transações da seguinte forma:

- Especificar todo o caso de uso, se o tempo necessário para essa tarefa for considerado aceitável;
- Caso contrário, deve-se montar a estrutura geral do caso de uso, especificando somente a(s) parte(s) dos casos de uso relacionada(s) à manutenção, ou seja, os passos do fluxo de eventos afetados com a manutenção. Isto é possível porque a TUCP-M leva em consideração somente as transações incluídas, alteradas e excluídas;

É possível ainda que seja necessário apresentar uma primeira estimativa logo no início do projeto, quando os casos de uso ainda não estão detalhados o suficiente. Nesse caso, é necessário que a contagem das transações seja realizada logo após a fase de Iniciação do projeto, quando o escopo da manutenção está definido, mas os casos de uso ainda não estão modificados. Nessa situação, é possível ter uma noção do impacto em cada caso de uso afetado, sem efetivamente realizar as modificações necessárias, porque apesar de não estarem detalhados é viável a realização de uma avaliação de impacto inicial, baseada no levantamento que já foi executado na etapa de Iniciação. Dessa forma, podem-se identificar as transações que serão incluídas, alteradas e excluídas em cada caso de uso.

- **Estimar o tamanho da manutenção**

Efetuar, a partir do total das transações contabilizadas para a manutenção dos casos de uso, da contagem dos atores e dos fatores de complexidade técnica, o cálculo da estimativa de tamanho da manutenção por caso de uso e por etapa do ciclo de vida.

- **Estimar a produtividade**

A produtividade deve ser indicada para cada etapa do ciclo de vida, baseada na estabilidade dos requisitos e na experiência da equipe. Deve-se levar em consideração que

para cada etapa do ciclo de vida do projeto o esforço maior é realizado por determinado papel[2] desempenhado por algum membro da equipe.

Assim, para a etapa de Requisitos o papel mais relevante é do analista de sistemas; para a etapa de Análise e Projeto o papel mais relevante pode ser do analista de sistemas ou do arquiteto, dependendo da instituição; para a etapa de codificação o maior esforço está relacionado aos desenvolvedores; e na etapa de Teste, o analista de teste representa o papel mais relevante.

A produtividade deve ser, portanto, indicada levando-se em consideração além da estabilidade dos requisitos envolvidos na solicitação de mudança, quem desempenhará esses papéis durante a execução de cada etapa do projeto.

- **Estimar o esforço do projeto**

Calcular o esforço do projeto de manutenção com base no tamanho, produtividade e fatores ambientais.

- **Estimar o esforço dos casos de uso**

Calcular o esforço do projeto de manutenção por caso de uso afetado e por etapa do ciclo de vida, acrescendo um fator de manutenção (f_{UC_Ei}), pertencente ao intervalo [0, 1], pois dependendo do tipo de manutenção (corretiva, adaptativa, perfectiva e preventiva), algumas etapas do ciclo de vida do projeto não necessitam ser executadas ou serão executadas parcialmente.

- **Estimar o prazo e o custo do projeto**

Calcular o prazo e custo do projeto de manutenção, baseado principalmente na quantidade e perfil das pessoas alocadas ao projeto e no tamanho estimado do projeto.

4.2.4 Apurar o Realizado

Apurar os valores efetivamente realizados de tamanho e esforço, objetivando comparar com os números estimados.

[2] O **papel** define o comportamento e as responsabilidades de um indivíduo ou de um conjunto de indivíduos que trabalham juntos como uma equipe, no contexto de uma organização de engenharia de software (RUP, 2003).

4.2.5 Comparar o Estimado com o Realizado

Calcular o percentual de erro relativo para cada caso de uso afetado por etapa do ciclo de vida e do projeto como um todo, por meio da comparação entre os valores estimados e os realizados.

4.2.6 Calibrar e Melhorar a Técnica

A comparação entre o estimado e o realizado, bem como a análise do que ocorreu durante o projeto e quais as causas que levaram ao percentual de erro relativo, proporcionam a avaliação constante da técnica e a calibragem necessária para o aumento de sua acurácia.

Na Seção 4.3 é detalhado o processo para calcular a estimativa de um projeto de manutenção de software, utilizando a TUCP-M.

4.3 Processo de Estimar Utilizando a TUCP-M

Com o recebimento da solicitação de mudança e após realizada a análise de impacto, a TUCP-M calcula de forma similar aos modelos paramétricos, inicialmente a estimativa de tamanho, em seguida estima o esforço e por último o prazo e o custo do projeto de manutenção.

4.3.1 Estimativa de Tamanho da Manutenção

A estimativa de tamanho na TUCP-M envolve as seguintes etapas:

- Contagem dos atores dos casos de uso afetados com a manutenção (UAW)
- Contagem dos casos de uso afetados com a manutenção (TUUCW)
- Cálculo dos pontos de casos de uso não ajustados (TUUCP)
- Cálculo dos fatores de complexidade técnica (TCF)
- Cálculo da estimativa de tamanho da manutenção (TUCP-M)

4.3.1.1 Contagem dos Atores

A maneira utilizada para classificar os atores envolvidos em cada caso de uso é semelhante à utilizada na técnica UCP (KARNER, 1993). Entretanto, devem ser considerados

para efeito de contagem apenas os atores afetados na manutenção, isto é, os atores dos casos de uso sensibilizados com o pedido de manutenção.

Os atores devem ser classificados quanto à sua complexidade, atribuindo-se a cada um deles um peso correspondente, segundo a Tabela 4.1.

O valor de UAW (*Unadjusted Actor Weight*) para um projeto de manutenção é, portanto, o somatório dos pesos dos atores dos casos de uso afetados com a manutenção.

Tabela 4.1 - Classificação dos atores (KARNER, 1993)

Tipo	Descrição	Peso
Simples	Aplicação com APIs definidas	1
Médio	Aplicação com interface baseada em protocolo ou interação de usuário baseado em linhas de comandos	2
Complexo	Interação de usuário através de interface gráfica ou página Web	3

4.3.1.2 Contagem dos Casos de Uso

- **Identificação e Contagem das Transações**

Para a realização da contagem dos casos de uso, deve-se primeiramente realizar a contagem das transações incluídas, alteradas e excluídas em cada caso de uso afetado.

A identificação das transações segue as mesmas orientações da TUCP e estão apresentadas no Apêndice B. Entretanto, durante a utilização da TUCP-M na organização onde foram realizados experimentos da técnica, pôde-se contribuir com algumas melhorias conforme relacionadas a seguir:

- O que deve ser contado como uma transação (itens aperfeiçoados):
  - Passos do fluxo de eventos do caso de uso que contenham uma geração de relatório são considerados como uma transação. A quantidade de filtros será a quantidade de transações, se para cada filtro for necessário acessar banco de dados distintos.
  - Fluxo alternativo do caso de uso que contiver uma quantidade de validações maior ou igual a 5, conta-se como uma transação. Se a quantidade de validações for maior que 10, conta-se uma transação a cada grupo de 5 validações. Ou seja, até 10 validações, conta-se 1 transação, a partir daí, conta-se 1 transação a cada grupo de 5 validações.
  - Passos do fluxo de eventos do caso de uso onde existirem validações simples (obrigatoriedade, formato, etc.) de campo de entrada de dados são considerados como uma única transação se a quantidade de validações for menor ou igual a 10.

Se a quantidade de validações for maior que 10, conta-se uma transação a cada grupo de 5 validações. Ou seja, até 10 validações, conta-se 1 transação, a partir daí, conta-se 1 transação a cada grupo de 5 validações.

- O que não deve ser contado como uma transação (item incluído):
  - Rotinas, casos de uso, cálculos ou qualquer componente que já estiverem prontos para reuso. Por exemplo: cálculo/validação de dígito verificador (módulo 10 ou módulo 11) ou o cálculo/validação do dígito do CPF/CNPJ não deve ser classificado como uma validação complexa e portanto contado como uma transação, quando esse procedimento já existir na organização.

As contagens das transações e das transações de manutenção estão exemplificadas no Apêndice C. Cada passo do caso de uso considerado como uma transação foi indicado por meio da representação "(**1T**)", no final de sua descrição. Cada transação que foi incluída, alterada ou excluída foi indicada por meio da representação "(**1TM**)", que indica uma transação de manutenção.

Sugere-se que seja adotada como padrão na organização a identificação das transações no documento de especificação de caso de uso como exemplificado no Apêndice C. Esta identificação facilita dentre outros:

- A documentação referente à contagem das transações utilizada para estimar o tamanho do projeto
- Revisão por pares da especificação dos casos de uso e da contagem das transações
- Negociação com empresas prestadoras de serviços terceirizados
- Recontagem das transações por algum motivo

- **Classificação dos Casos de Uso**

Devem ser considerados para efeito de contagem, apenas os casos de uso afetados na manutenção, isto é, aqueles que sofreram algum tipo de modificação em virtude das mudanças solicitadas. Eles devem ser classificados quanto à sua complexidade, atribuindo-se a cada um deles um peso correspondente, segundo a Tabela 4.2.

Finalizada a apuração da quantidade de transações requeridas para inclusão (t_i), alteração (t_a), ou exclusão (t_e) de passos do fluxo de eventos de cada caso de uso, que foi

afetado pela manutenção, pode-se obter o total de transações de manutenção para cada caso de uso, conforme mostrado na Equação 4.1:

$$t_m = \Sigma\, t_i + \Sigma\, t_a + \Sigma\, t_e \qquad (Eq.\ 4.1)$$

O valor de t_m de cada caso de uso determina sua complexidade conforme Tabela 4.2, classificando-o como simples, médio, complexo e *n*-complexo.

Tabela 4.2 - Classificação dos casos de uso

Tipo	Descrição	Peso
Simples	Até 3 transações de manutenção, incluindo os passos alternativos	5
Médio	De 4 a 7 transações de manutenção, incluindo os passos alternativos	10
Complexo	De 8 a *t* transações de manutenção, incluindo os passos alternativos	15
n-complexo	Acima de *t* transações de manutenção	P_X

Os casos de uso com até 3 transações de manutenção são calculados da mesma maneira apresentada por Karner (1993). Acima de *t* transações de manutenção são calculados conforme a TUCP, apresentado na Seção 3.3.7.

4.3.1.3 Cálculo dos Pontos de Casos de Uso não Ajustados

O cálculo dos pontos de caso de uso técnicos não ajustados, TUUCP (*Technical Unadjusted Use Case Points*), é efetuado pelo somatório entre a complexidade dos atores e o cálculo da complexidade dos casos de uso, conforme Equação 4.2 (MONTEIRO, 2005).

$$\text{TUUCP} = \Sigma\, \text{UAW} + \Sigma\, \text{TUUCW} \qquad (Eq.\ 4.2)$$

4.3.1.4 Cálculo dos Fatores de Complexidade Técnica

Os Fatores de complexidade técnica, TCF (*Technical Complexity Factor)* são fatores de ajuste para o cálculo da estimativa do tamanho do projeto, e relacionam-se com seus requisitos funcionais. O cálculo é realizado da mesma forma que a técnica original UCP. Os Fatores de Complexidade Técnica são os mesmos utilizados pela UCP e pela TUCP. São exibidos novamente na Tabela 4.3 com seus respectivos pesos, apenas para facilitar a leitura desse trabalho.

Tabela 4.3 - Fatores de complexidade técnica (Ribu, 2001)

Fator	Descrição	Peso	Nota
T1	Sistemas Distribuídos	2.0	
T2	Desempenho da Aplicação	1.0	
T3	Eficiência do Usuário-Final (*on-line*)	1.0	
T4	Processamento Interno Complexo	1.0	
T5	Reusabilidade do código em outras aplicações	1.0	
T6	Facilidade de Instalação	0.5	
T7	Usabilidade (facilidade operacional)	0.5	
T8	Portabilidade	2.0	
T9	Facilidade de Manutenção	1.0	
T10	Concorrência	1.0	
T11	Características especiais de segurança	1.0	
T12	Acesso direto para terceiros	1.0	
T13	Facilidades especiais de treinamento	1.0	

De acordo com a UCP, os Fatores de Complexidade Técnica podem receber notas que variam de 0 a 5, o valor 0 (zero) indicando que este item é irrelevante para o projeto, de pouca criticidade e baixa complexidade; o valor 3 indicando influência moderada; e o valor 5 uma forte influência, alta criticidade e complexidade.

Para facilitar a avaliação de cada um desses fatores, propõe-se a utilização da Tabela 4.4 de termos lingüísticos.

Tabela 4.4 – Termos lingüísticos para avaliação dos Fatores de Complexidade Técnica (FREIRE & BELCHIOR, 2007c)

Nota	Termo lingüístico
0	Sem influência ou relevância
1	Fraca influência ou relevância
2	Moderada influência ou relevância
3	Média influência ou relevância
4	Alta influência ou relevância
5	Total influência ou relevância

O TCF é obtido por meio da Equação 4.3 (KARNER, 1993), onde TFactor corresponde ao somatório dos produtos entre o peso e a nota atribuída de cada um dos fatores de complexidade técnica:

$$TCF = 0{,}6 + (0{,}01 * TFactor) \qquad (Eq.\ 4.3)$$

4.3.1.5 Cálculo da Estimativa de Tamanho da Manutenção

No cálculo do tamanho do software na UCP, os fatores de ambiente (EF) são considerados. Segundo a TUCP (MONTEIRO, 2005), o uso desses fatores pode levar a

diferentes tamanhos, dependendo da equipe ou empresa que desenvolver um dado projeto. Assim, entende-se que deve ser evitado gerar dependência do tamanho com características da equipe de desenvolvimento, tais como motivação, familiaridade com o processo de desenvolvimento e experiência profissional. Todavia, os fatores ambientais têm sua influência efetiva no cálculo das estimativas de esforço.

Dessa forma, os fatores técnicos ambientais (EF) devem ser desatrelados do cálculo do tamanho, uma vez que o tamanho é uma grandeza física e não deve ter seu valor alterado em função de fatores ambientais. Neste contexto, os fatores ambientais (EF) não serão levados em consideração para efeito de cálculo na estimativa de tamanho da TUCP-M, mas serão considerados no cálculo das estimativas de esforço da manutenção.

O cálculo da estimativa de tamanho de manutenção, isto é a TUCP de manutenção (TUCP-M), que correspondente aos pontos de caso de uso técnicos ajustados, é dado pela Equação 4.4:

$$TUCP - M = TUUCP * TCF \qquad (Eq.\ 4.4)$$

Em um projeto de manutenção, considerando-se suas *n* etapas de ciclo de vida (definidas no processo de software), o tamanho de um projeto de manutenção (TUCP-M) corresponde ao total do tamanho da manutenção realizada em cada uma de suas *n* etapas (*E*), de acordo com a Equação 4.5:

$$TUCP - M = \sum_{i=1}^{n} TUCP - M_{Ei} \qquad (Eq.\ 4.5)$$

Esse tamanho também corresponde ao somatório de cada um de seus *m* casos de uso (UC), conforme Equação 4.6:

$$TUCP - M = \sum_{j=1}^{m} TUCP - M_{UCj} \qquad (Eq.\ 4.6)$$

4.3.2 Estimativa de Tamanho da Manutenção por Etapa do Ciclo de Vida

Assim como em projetos de desenvolvimento, estimar o tamanho da manutenção por etapa do ciclo de vida proporciona a obtenção de uma granularidade maior da estimativa, tornando mais fácil o acompanhamento do projeto e permitindo ações corretivas antes do final de cada etapa do ciclo de vida desse projeto.

Como o esforço de um projeto é diretamente proporcional a seu tamanho, então o tamanho de um caso de uso por etapa do ciclo de vida pode ser baseado no percentual de

esforço empregado para seu desenvolvimento. Assim sendo, e baseado no trabalho de MENESES (2001) *apud* MONTEIRO (2005), definiu-se a estimativa de tamanho do caso de uso por etapa do ciclo de vida na Equação 4.7:

$$TUCP - M_{(UC\_Etapa)} = \left(\left(\frac{TUCP - M}{\sum TUUCW} \right) * TUUCW_{(UC)} \right) * \mu_E \qquad (Eq.\ 4.7)$$

TUCP-$M_{(UC\_Etapa)}$ é o tamanho do caso de uso por etapa do ciclo de vida, sendo definido proporcionalmente a seu fator de complexidade e ao percentual de distribuição de esforço para a etapa do ciclo de vida considerada. O $TUUCW_{UC}$ é o peso do caso de uso que está sendo calculado. O valor de μ_E corresponde ao percentual de realização estimado para a atividade (ou etapa) em mensuração.

O tamanho do caso de uso em TUCP-M como um todo ao longo do projeto é mostrado na Equação 4.8:

$$TUCP - M_{(UC)} = \left(\frac{TUCP - M}{\sum TUUCW} \right) * TUUCW_{(UC)} \qquad (Eq.\ 4.8)$$

A partir dos dados de projetos já concluídos da organização, podem-se levantar percentuais de realização (μ_E) de cada etapa do ciclo de vida (E), que em geral seguem uma média histórica própria da organização. Caso a organização não tenha um histórico de seus projetos, pode-se partir de pesquisa na literatura, ou em outras organizações, do percentual utilizado e ir ajustando tais percentuais à medida que novos projetos são realizados. Assim, para se estimar o tamanho da manutenção de determinada etapa do ciclo de vida, deve-se multiplicar o tamanho estimado do projeto pelo percentual de realização da etapa. A representação dessa fórmula pode ser dada pela Equação 4.9:

$$TUCP - M_E = \mu_E * TUCP - M \qquad (Eq.\ 4.9)$$

Nesse trabalho, consideram-se as seguintes etapas do ciclo de vida de software, por serem consideradas as mais representativas: (*i*) Requisitos (*Req*); (*ii*) Análise e Projeto (*A&P*); (*iii*) Codificação (*Cod*); e (*iv*) Teste (*Tst*).

Nos exemplos apresentados no Capítulo 5, percebe-se que na organização onde o experimento foi realizado, iniciou-se com uma outra forma de determinação das etapas. Conforme o amadurecimento da organização, as etapas foram redefinidas.

4.3.3 Estimativa de Esforço de Manutenção

A estimava de esforço de manutenção envolve as seguintes etapas:

- Cálculo dos fatores ambientais (EF)
- Cálculo da produtividade (Prod)
- Cálculo do esforço do projeto e do esforço por etapa do ciclo de vida

4.3.3.1 Cálculo dos Fatores Ambientais

Os Fatores ambientais *EF (Environmental Factors)* são fatores de ajuste para o cálculo da estimativa de esforço, e relacionam-se com os requisitos não-funcionais do projeto. São calculados conforme apresentados pela técnica UCP e são os mesmos utilizados pela UCP e pela TUCP. São exibidos novamente na Tabela 4.5 com seus respectivos pesos, apenas para facilitar a leitura desse trabalho.

Tabela 4.5 - Fatores ambientais (RIBU, 2001)

Fator	Descrição	Peso	Nota
F1	Familiaridade com o processo de desenvolvimento	**1,5**	
F2	Experiência com a aplicação em desenvolvimento	**0,5**	
F3	Experiência com orientação a objeto	**1**	
F4	Capacidade de análise	**0,5**	
F5	Motivação	**1**	
F6	Requisitos estáveis	**2**	
F7	Colaboradores de meio período	**-1**	
F8	Dificuldade na linguagem de programação	**-1**	

O EF é obtido pela Equação 4.10 (KARNER, 1993), onde EFator corresponde ao somatório dos produtos entre o peso e a nota atribuída a cada um dos fatores ambientais.

$$EF = 1{,}4 + (-0{,}03 * EFator) \qquad (Eq.\ 4.10)$$

Para os fatores de F1 a F4, 0 (zero) significa nenhuma experiência no assunto, 5 significa excelência no assunto e 3 um conhecimento mediano sobre o assunto. Para o fator F6, 0 significa requisitos instáveis, 5 imutáveis e 3 poucas mudanças. Para o fator F7, 0 significa nenhum colaborador por meio período, 5 significa que todos os colaboradores trabalham meio período e 3 que alguns colaboradores trabalham meio período. Para o fator F8, 0 significa pouca ou nenhuma dificuldade na linguagem de programação, 3 significa uma complexidade média e 5 uma alta complexidade e dificuldade.

Objetivando facilitar a avaliação de cada um desses fatores, propõe-se a utilização da Tabela 4.6 de termos lingüísticos:

Tabela 4.6 – Termos lingüísticos para avaliação dos EFs (FREIRE & BELCHIOR, 2007c)

Nota	Termo Lingüístico
0	Não
1	Pouco (a)
2	Moderado (a)
3	Médio (a)
4	Alto (a)
5	Total

4.3.3.2 Cálculo da Produtividade

O fator de produtividade (Prod) está diretamente relacionado com a estabilidade dos requisitos e a experiência da equipe do projeto, que pode ser obtido a partir de resultados de projetos semelhantes já concluídos e armazenados em uma base histórica da organização.

Entretanto, utilizar esse valor histórico muitas vezes não é a forma mais adequada, pois a equipe de um projeto de manutenção nem sempre é a mesma que desenvolveu ou já realizou outros projetos de manutenção semelhantes. Além disso, cada projeto de manutenção de um mesmo produto de software não possui o mesmo grau de estabilidade de requisitos.

Nesse trabalho, serão utilizados os valores de referência para a produtividade propostos por Anda *et al.* (2001) onde o esforço pode variar entre 15 h.h por ponto de caso de uso para projetos com requisitos estáveis e com equipe experiente, e 30 h.h por ponto de caso de uso para projetos com requisitos não estáveis e com equipe não experiente.

Utilizar, por exemplo, o fator de produtividade entre 15 e 30, pode significar até duplicar o esforço do projeto. Para calibrar o fator de produtividade mais adequado para o projeto de manutenção, de forma menos subjetiva, combinando experiência da equipe e estabilidade de requisitos, sugere-se que o cálculo seja realizado a partir dos seguintes fatores ambientais (*EF*):

- Para a Experiência da equipe de projeto:
  - **F1**: Familiaridade com o processo de desenvolvimento. Cujo peso correspondente na Tabela 4.5 é igual a 1,5;
  - **F2**: Experiência com a aplicação em desenvolvimento. Cujo peso correspondente na Tabela 4.5 é igual a 0,5;
  - **F3**: Experiência com orientação a objeto. Cujo peso correspondente na Tabela 4.5 é igual a 1;
- Para a Estabilidade dos requisitos:
  - **F6:** Requisitos estáveis. Cujo peso correspondente na Tabela 4.5 é igual a 2;

Considerando-se os fatores ambientais F1, F2, F3 e F6, tem-se então:

$x_1 = (1{,}5\ F1 + 0{,}5F2 + F3) \rightarrow$ variável experiência da equipe

$x_2 = 2\ F6 \rightarrow$ variável estabilidade dos requisitos

$x = x_1 + x_2$

Para $F1 = F2 = F3 = F6 = 0$, tem-se a produtividade mais baixa (b), isto é, equipe sem experiência e requisitos não estáveis. Neste caso, $x = 0$, Prod = b, $b = 30$.

Para $F1 = F2 = F3 = F6 = 5$, tem-se a produtividade mais alta (a), isto é, equipe experiente e requisitos estáveis. Neste caso, $x = 25$, Prod = a, $a = 15$.

A partir desses dados, pode-se calcular a produtividade (Prod), utilizando-se a equação linear abaixo:

$$\text{Prod} = b - [(b - a) / 25]\ x$$

Como nesse trabalho a produtividade varia de 15 a 30, então $a = 15$ e $b = 30$, tem-se a Equação 4.11:

$$\text{Prod} = 30 - 0{,}6\ x \qquad (Eq.\ 4.11)$$

Considerando-se que a experiência da equipe (x_1) de projeto está diretamente relacionada ao papel desempenhado em uma etapa do ciclo de vida (E), pode-se então calcular a produtividade a partir desses papéis. Assim sendo, ter-se-ía a variável experiência da equipe alocada àquela etapa de ciclo de vida (e.g.: experiência dos analistas de sistemas, experiência dos arquitetos, etc.).

Considerando-se também que a variável estabilidade de requisitos (x_2) pode variar por caso de uso, que agrupa o comportamento de um subconjunto de funcionalidades (requisitos), pode-se então calcular a produtividade a partir da estabilidade dos requisitos de cada caso de uso.

A produtividade do projeto (Prod) pode ser dada pela média ponderada da produtividade das etapas de seu ciclo de vida (Prod_E), considerando-se o tamanho de cada etapa (TUCP-M_E), como mostrado na Equação 4.12.

$$\text{Pr}od = \sum_{i=1}^{n} (\text{Pr}od_{Ei} * TUCP - M_{Ei}) / TUCP - M \qquad (Eq.\ 4.12)$$

A produtividade do projeto (Prod) pode ser dada pela média ponderada da produtividade dos casos de uso (Prod_{UC}), considerando-se o tamanho de cada caso de uso (TUCP-M_{UC}), conforme Equação 4.13.

$$\text{Pr}od = \sum_{j=1}^{n} (\text{Pr}od_{UCj} * TUCP - M_{UCj}) / TUCP - M \qquad (Eq.\ 4.13)$$

4.3.3.3 Cálculo do Esforço do Projeto e do Esforço por Etapa do Ciclo de Vida

De forma semelhante ao cálculo do esforço de um projeto de software na TUCP, o esforço de um projeto de manutenção na TUCP-M, pode ser obtido baseado no tamanho do projeto de manutenção, em seus fatores ambientais e na produtividade calculada, como mostra a Equação 4.14:

$$Esforço = TUCP - M * EF * \text{Pr}od \qquad (Eq.\ 4.14)$$

O esforço referente a cada etapa do ciclo de vida do projeto ($Esforço_E$) pode ser calculado conforme Equação 4.15:

$$Esforço_E = TUCP - M_E * EF * \text{Pr}od_E \qquad (Eq.\ 4.15)$$

E o esforço do caso de uso por etapa do ciclo de vida do projeto pode ser calculado conforme Equação 4.16:

$$Esforço_{UC\_E} = TUCP - M_{UC\_E} * EF * \text{Pr}od_E \qquad (Eq.\ 4.16)$$

A estimativa de esforço do projeto de manutenção (*Esforço*) corresponde ao esforço total realizado em suas *n* etapas do ciclo de vida do projeto (*E*), utilizando a Equação 4.17:

$$Esforço = \sum_{i=1}^{n} Esforço_{Ei} \qquad (Eq.\ 4.17)$$

O esforço de um projeto de software pode ser dado também pelo somatório do esforço para a manutenção de seus *m* casos de uso (*UC*), conforme equação 4.18:

$$Esforço = \sum_{j=1}^{m} Esforço_{UCj} \qquad (Eq.\ 4.18)$$

Para se calcular o esforço referente a cada caso de uso afetado ($Esforço_{UC}$), pode-se utilizar a Equação 4.19:

$$Esforço_{UC} = \sum_{i=1}^{n} Esforço_{UC\_Ei} \qquad (Eq.\ 4.19)$$

Entretanto, em um projeto de manutenção, etapas do ciclo de vida (*E*) do projeto podem ser parcialmente executadas. Por exemplo:

- Nas manutenções corretivas onde são detectados problemas de implementação. Nesse caso, pouco ou nenhum esforço é necessário para as etapas de requisitos e de análise e projetos. Somente será necessário o emprego de esforço na etapa de codificação e na etapa de teste.
- Nas manutenções adaptativas, a probabilidade maior é que seja necessário um esforço maior nas etapas de análise e projeto, codificação e teste.
- Nas manutenções perfectivas e preventivas, provavelmente todas as etapas do ciclo de vida necessitarão de esforço de adaptação.

Dessa forma, o esforço de manutenção de cada etapa do ciclo de vida do projeto de manutenção, para cada caso de uso, deve ser acrescido de um fator de manutenção (f_{UC_Ei}), pertencente ao intervalo [0, 1]. Quando $f_{UC\_E} = 0$, significa que o esforço de manutenção da etapa considerada para o caso de uso em questão é nulo e, quando $f_{UC\_E} = 1$, o esforço de manutenção é de 100%. O novo esforço de manutenção do caso de uso ($Esforço'_{UC}$) é dado pela Equação 4.20:

$$Esforço'_{UCj} = \sum_{i=1}^{n} (f_{UCj\_Ei} * Esforço_{UCj\_Ei}) \qquad (Eq.\ 4.20)$$

Lembrando que o esforço do caso de uso por etapa do ciclo de vida pode ser calculado levando-se em consideração o percentual de realização (μ_E) de cada etapa do ciclo de vida, definido para cada organização, tem-se a Equação 4.21:

$$Esforço_{UCj\_Ei} = \mu_{Ei} * Esforço_{UCj} \qquad (Eq.\ 4.21)$$

O novo esforço de manutenção do caso de uso por etapa do ciclo de vida ($Esforço'_{UC\_E}$) pode ser dado pela Equação 4.22:

$$Esforço'_{UCj\_Ei} = Esforço_{UCj} * \mu_{Ei} * f_{UCj\_Ei} \qquad (Eq.\ 4.22)$$

O novo esforço de manutenção por etapa do ciclo de vida ($Esforço'_{Ei}$) é dado pela Equação 4.23:

$$Esforço'_{Ei} = \sum_{j=1}^{m} Esforço'_{UCj\_Ei} \qquad (Eq.\ 4.23)$$

O novo esforço de um projeto de manutenção de software (*Esforço'*), que utiliza casos de uso, poderá ser expresso pela Equação 4.24:

$$Esforço' = \sum_{i=1}^{m} Esforço'_{Ei} \qquad (Eq.\ 4.24)$$

4.3.4 Estimativas de Prazo e Custo

As estimativas de prazo podem ser obtidas a partir do esforço estimado na Seção 4.3.3.3, disponibilidade de recursos, restrições do projeto, entre outras questões. O uso de técnicas do tipo PERT (*Programme Evaluation Review Technique*) e CPM (*Critical Path Method)* pode auxiliar na determinação mais precisa do prazo do projeto de manutenção, levando em consideração caminhos críticos, alocação de recursos, seqüenciamento de atividades, e dependência de tarefas.

A estimativa de custo do projeto pode ser determinada utilizando-se a Equação 4.25, em que o "Valor" corresponde ao desembolso monetário realizado para uma unidade de tamanho.

$$\text{Custo}_{(\text{unidade\_monetária})} = \text{TUCP - M} * \text{Valor}_{(\text{TUCP - M})} \qquad (Eq.\ 4.25)$$

O custo calculado pela *Eq*. 4.25 pode ser, por exemplo, utilizado em contratos de terceirização de serviços de fábrica de software. É feita uma estimativa de tamanho do projeto de manutenção e esse valor é multiplicado pelo valor que foi contratado para cada TUCP-M.

Uma outra forma de cálculo do custo é utilizada, por exemplo, para contratos de terceirização de mão-de-obra de análise, projeto, codificação e teste de um projeto de manutenção. Em geral, uma etapa do ciclo de vida é realizada mais fortemente por um determinado papel. Por exemplo, na etapa de "Requisitos", o analista de sistemas desempenha o papel mais importante na equipe do projeto. Assim sendo, tendo-se o valor médio por hora pago aos analistas de sistemas, pode-se calcular o custo da etapa "Requisitos" a partir do esforço estimado para essa etapa, conforme apresentado na Equação 4.26:

$$\text{Custo}_{Ei(\text{unidade\_monetária})} = \text{Esforço}_{Ei} * \text{Valor}_{(\text{papel})} \qquad (Eq.\ 4.26)$$

Baseado no cronograma do projeto, em geral elaborado a partir da WBS (*Work Breakdown Structure*), onde estão distribuídos todos os esforços por papéis, pode-se estimar de forma mais acurada o custo do projeto de manutenção, com base no valor médio por hora pago a cada profissional envolvido, conforme mostrado na Equação 4.27:

$$Custo_{(unidade\_monetária)} = \sum_{i=1}^{n} Custo_{Ei} \qquad (Eq.\ 4.27)$$

Semelhante à TUCP o custo aqui calculado está relacionado apenas ao esforço dos casos de uso do projeto.

4.3.5 Cálculo do Erro Relativo Simétrico

Para se avaliar a precisão da estimativa calculada e portanto a viabilidade de sua utilização, deve-se fazer uma comparação entre o esforço estimado e o esforço real do projeto.

No caso da TUCP-M, após a finalização de cada projeto calcula-se o seu erro relativo simétrico (*ERS*) conforme Equação 4.28 e Equação 4.29. Para o cálculo do percentual de erro estimado do esforço por etapa do ciclo de vida e para todo o projeto, é utilizada a métrica *Symmetric Relative Error* (*SER*), proposta por M. Jorgensen e D. Sjobeg *apud* Ribu (2001), onde a partir do esforço estimado e esforço real do projeto, pode ser calculado seu erro relativo simétrico (ERS), em que:

$$ERS = (Real - Estimado) / Real, \text{ se Real} \leq \text{Estimado} \quad (Eq.\ 4.28)$$

$$ERS = (Real - Estimado) / Estimado, \text{ se Real} \geq \text{Estimado} \quad (Eq.\ 4.29)$$

De acordo com o gráfico apresentado na Figura 3.3, o percentual de erro aceitável para as estimativas diminui à medida que as etapas do ciclo de vida do projeto são executadas. O cálculo do erro relativo simétrico (*ERS*) após a finalização de cada projeto de manutenção ou ao final de cada iteração do projeto, permite avaliar se os valores apresentados estão dentro destes limites aceitáveis, além de verificar sua acurácia, analisando a necessidade ou não de calibrações, objetivando a obtenção de resultados cada vez mais realistas e o aprimoramento das estimativas realizadas com a TUCP-M.

Na seção 4.4, é apresentado um exemplo da aplicação da TUCP-M para um projeto de manutenção de software.

4.4 Exemplo de Aplicação da TUCP-M

A aplicação da técnica em projetos de manutenção é apresentada no Capítulo 5, cada um abrangendo vários casos de uso, bem como a evolução e calibração da TUCP-M durante seu período de utilização. Entretanto, o objetivo principal dessa seção é exemplificar de forma simples e o mais direta possível a utilização da técnica TUCP-M para estimar o tamanho, esforço, prazo e custo de um projeto de manutenção de software. Uma planilha eletrônica foi construída para auxiliar nos cálculos necessários para apuração do tamanho, esforço e prazo de todos os projetos apresentados nesse trabalho.

Para o exemplo mencionado, será utilizado um projeto de manutenção, realizado em uma instituição financeira de grande porte, doravante denominado de Projeto Y. A solicitação de mudança relacionada afetou quatro casos de uso do sistema denominado Risco de Crédito.

O sistema Risco de Crédito foi desenvolvido em ambiente *Windows* e WEB, com arquitetura três camadas, utilizando as linguagens de programação ASP.Net, VB e C#. A base de dados foi construída parte utilizando banco de dados DB2 e parte utilizando o SQL-Server. O sistema objetiva calcular o risco das operações de crédito da organização, baseado em resoluções do Banco Central do Brasil, classificar essas operações de crédito quanto ao risco, calcular o limite de risco dos clientes da organização e enviar informações para o Banco Central.

O Projeto Y contou com 1 analista de sistemas, um programador, e 1 analista de teste. Para exemplificar a utilização da TUCP-M serão acompanhados apenas os casos de uso UC_01 – Calcular LRC Automático e UC_28 – Disponibilizar Portfólio.

O documento de solicitação de mudança preenchido é apresentado no Apêndice D (Solicitação de Mudança para o Sistema Risco de Crédito) e a especificação de um dos casos de uso em questão é mostrada no Apêndice C (Especificação de Caso de Uso do Sistema de Risco de Crédito). Na especificação do caso de uso, são destacadas as transações existentes como (**1T**) e as transações que se originaram da manutenção solicitada como (**1TM**), que são as transações referentes ao projeto de manutenção.

Após recebimento da solicitação de mudança, os requisitos devem ser identificados e os casos de uso afetados devem ser mapeados. Para cada caso de uso afetado, deve ser realizada a contagem das transações incluídas, alteradas e excluídas. Feito isso, inicia-se o cálculo do tamanho do Projeto Y, que passa pelas seguintes etapas:

4.4.1 Contagem dos atores (UAW)

O sistema possui 9 atores, dois deles complexos e os demais simples. Entretanto, os casos de uso afetados com a manutenção e escolhidos para exemplo possuem os atores classificados como apresentado na Tabela 4.7. Pode-se calcular o peso dos atores conforme Tabela 4.1 de classificação dos atores.

O valor do peso dos atores para esta manutenção é a soma do peso total de cada um dos atores, ou seja: UAW = 7.

Tabela 4.7 – Peso dos atores do Projeto Y

Ator	Complexidade	Peso
Gerente de Agência	Complexo	3
Administrador	Complexo	3
Sistema Alfa	Simples	1
	Total	7

4.4.2 Contagem dos casos de uso (TUUCW)

Para o Projeto Y foi verificado que:

- O caso de uso UC_01 - Calcular LRC Automático já existia e foi necessário alterar 3 transações em virtude da manutenção solicitada. Apesar deste caso de uso possuir 5 transações, para o Projeto Y será levado em consideração para a contagem deste caso de uso apenas as três transações alteradas.
- Foi necessário implementar um novo caso de uso, o UC_28 – Disponibilizar Portfólio com 7 transações.

A contagem dos casos de uso para o Projeto Y é apresentada na Tabela 4.8. Usaram-se como referência os pesos mencionados na Tabela 4.2.

Tabela 4.8 – Contagem dos casos de uso do Projeto Y

Caso de Uso	Qtde. Transações na manutenção	Complexidade	TUUCW
UC_01	3	Simples	5
UC_28	7	Médio	10
Total	10	-	15

A complexidade dos casos de uso é obtida pela soma de seus pesos: TUUCW = 15.

4.4.3 Cálculo dos pontos de casos de uso não ajustados (TUUCP)

O cálculo de TUUCP do Projeto Y é similar ao cálculo dos pontos de caso de uso não ajustados (*UUCP- Unadjusted Use Case Points*) da técnica UCP. Basta somar o cálculo da complexidade de atores com o cálculo da complexidade dos casos de uso, conforme a *Eq.* 4.2.

$$TUUCP = \Sigma\ UAW + \Sigma\ TUUCW \qquad (Eq.\ 4.2)$$

$$TUUCP = 7 + 15 = 22$$

4.4.4 Cálculo fatores de complexidade técnica (TCF)

No exemplo apresentado, a Tabela 4.9 representa o valor indicado para cada um dos fatores de complexidade técnica do Projeto Y. Esses valores ou notas foram dados conforme os termos lingüísticos mencionados na Tabela 4.4.

Tabela 4.9 – Cálculo dos fatores de complexidade técnica - Projeto Y

Fator	Descrição	Peso	Valor	Produto
T1	Sistemas distribuídos	2,0	0	0
T2	Desempenho da aplicação	1,0	4	4
T3	Eficiência do usuário final (on-line)	1,0	3	3
T4	Processamento interno complexo	1,0	3	3
T5	Reusabilidade do código em outras aplicações	1,0	0	0
T6	Facilidade de instalação	0,5	2	1
T7	Usabilidade (facilidade operacional)	0,5	3	1,5
T8	Portabilidade	2,0	3	6
T9	Facilidade de manutenção	1,0	0	0
T10	Concorrência	1,0	0	0
T11	Características especiais de segurança	1,0	0	0
T12	Acesso direto para terceiros	1,0	0	0
TFator				**18,5**

O cálculo do TFator é dado por:

TFator = Σ(Peso * Valor)

TFator = (2*0) + (1*4) + (1*3) + (1*3) + (1*0) + (0,5 * 2) + (0,5 * 3) + (2 * 3) +
(1 * 0) + (1 * 0) + (1 * 0) + (1 * 0)

TFator = 18,5

Conforme *Eq.* 4.3 calcula-se o valor dos Fatores de Complexidade Técnica:

TCF = 0,6 + (0,01 * TFactor)

TCF = 0,6 + (0,01 * 18,5)

TCF = 0,785

4.4.5 Cálculo dos pontos de caso de uso técnico para manutenção

Da *Eq.* 4.4 chega-se ao tamanho do projeto de manutenção, ou seja, ao cálculo dos pontos de caso de uso técnicos para manutenção (TUCP-M):

TUCP-M = TUUCP * TCF

TUCP-M = 22 * 0,785

TUCP-M = 17,27

Para calcular o tamanho da manutenção por etapa do ciclo de vida utilizou-se a *Eq.* 4.9. No caso da instituição onde o exemplo foi aplicado, utilizam-se os seguintes percentuais por etapa do ciclo de vida:

Requisito: $\mu_{Req} = 20\%$

Análise e Projeto: $\mu_{A\&P} = 25\%$

Codificação: $\mu_{Cod} = 40\%$

Teste: $\mu_{Tst} = 15\%$

TUCP-M = 17,27

$$TUCP - M_E = \mu_E * TUCP - M \qquad (Eq.\ 4.9)$$

Assim, apresenta-se na Tabela 4.10 o tamanho de cada etapa do ciclo de vida para o Projeto Y.

Tabela 4.10 – TUCP-M por etapa do ciclo de Vida - Projeto Y

Etapa	TUCP-M
Requisito	3,45
Análise e Projeto	4,32
Codificação	6,91
Teste	2,59
Total Manutenção	17,27

Para calcular o tamanho da manutenção por caso de uso utilizou-se a *Eq*. 4.8.

$$TUCP - M_{(UC)} = \left(\frac{TUCP - M}{\sum TUUCW} \right) * TUUCW_{(UC)}$$

Onde:

TUCP-M = 17,27

$TUUCW_{UC\_01} = 5$ e $TUUCW_{UC\_28} = 10$

$\Sigma TUUCW = 15$

Dessa forma, o Tamanho do caso de uso UC_01 é de 5,76 TUCP-M e o do UC_28 é de 11,51 TUCP-M.

Para calcular o tamanho do caso de uso por etapa do ciclo de vida. Utilizou-se a *Eq*. 4.7. Ou seja, foi aplicado o percentual de cada etapa ao tamanho do caso de uso:

$$TUCP - M_{(UC\_Etapa)} = \left(\left(\frac{TUCP - M}{\sum TUUCW} \right) * TUUCW_{(UC)} \right) * \mu_E$$

Levando-se em consideração os percentuais das etapas ($\mu_{Req=}$20%, $\mu_{A\&P=}$25%, $\mu_{Cod=}$40%, $\mu_{Tst=}$15%), o tamanho do projeto (TUCP-M=17,27), e a contagem dos casos de uso (UC_01=5 e UC_28=10), a Tabela 4.11 apresenta os valores calculados para os casos de uso UC_01 e UC_28 por etapa do ciclo de vida:

Tabela 4.11 – Estimativa de tamanho dos Casos de Uso por etapa do ciclo de vida–Projeto Y

Caso de Uso	Requisito	Análise e Projeto	Codificação	Teste	Total
UC_01	1,15	1,44	2,30	0,86	5,76
UC_28	2,30	2,88	4,61	1,73	11,51

4.4.6 Cálculo dos fatores ambientais

Os fatores ambientais (EF) influenciam o cálculo do esforço do projeto. Na Tabela 4.12 apresentam-se estes fatores com seus respectivos pesos. Os valores ou notas foram dados conforme os termos lingüísticos mencionados na Tabela 4.6:

Tabela 4.12 – Fatores Ambientais - Projeto Y

Fator	Descrição	Peso	Valor	Peso Total
F1	Familiaridade com o processo de desenvolvimento	1,5	2	3
F2	Experiência com a aplicação em desenvolvimento	0,5	3	1,5
F3	Experiência com orientação a objeto	1	4	4
F4	Capacidade de análise	0,5	4	2
F5	Motivação	1	4	4
F6	Requisitos estáveis	2	4	8
F7	Colaboradores de meio período	-1	0	0
F8	Dificuldade na linguagem de programação	-1	0	0
EFator		**22,5**		

O cálculo dos fatores ambientais é dado por:

EF = 1,4 + (- 0,03 * EFator), onde EFator = $\sum$ (Peso * Valor). Assim,

EFator = (1,5*2) + (0,5*3) + (1*4) + (0,5*4) + (1*4) + (2*4) + (-1*0) + (-1*0)

EFator = 22,5

EF = 1,4 + (- 0,03 * 22,5)

EF = 0,725

4.4.7 Cálculo da produtividade

Pode-se obter a produtividade do projeto de manutenção de duas formas:

- Atribuindo valores de 15 a 30 para cada uma das etapas do ciclo de vida.
  - No exemplo apresentado, foi atribuído o valor 18 para a produtividade de cada uma das etapas do ciclo de vida.
- Ou utilizando a Eq. 4.11, para cada etapa do ciclo de vida, onde $Prod = 30 - 0{,}6\ x$

  $x = x_1 + x_2$

  $x_1 = (1{,}5\ F1 + 0{,}5F2 + F3) \rightarrow$ variável experiência da equipe

  $x_1 = (1{,}5*2) + (0{,}5*3) + 4$

  $x_1 = 8{,}5$

  $x_2 = 2\ F6 \rightarrow$ variável estabilidade dos requisitos

  $x_2 = 2*4$

$x_2 = 8$

$x = x_1 + x_2$

$x = 16{,}5$

$Prod = 30 - 0{,}6\ x$

$Prod = 20$

Na Seção 4.4.8 é mostrada a diferença na apuração do esforço do Projeto Y com a utilização dos dois valores calculados para a produtividade: 18 e 20.

4.4.8 Cálculo da estimativa de esforço

Para calcular o esforço do Projeto Y, levou-se em consideração o que se segue:

i. Foram utilizadas as equações referenciadas no item 4.3.3.3 desse trabalho, para calcular o esforço por etapa do ciclo de vida.

ii. O tamanho de cada etapa do ciclo de vida, como mostrado na Tabela 4.10: TUCP-M_{Req} = 3,45; TUCP-$M_{A\&P}$ = 4,32; TUCP-M_{Cod}= 6,91; TUCP-M_{Tst}= 2,59.

iii. Os percentuais de realização das etapas de requisitos, análise e projeto, codificação e teste na organização onde o projeto foi executado: μ_{Req} = 0,20 , $\mu_{A\&P}$ = 0,25 , μ_{Cod} = 0,40 , μ_{Tst} = 0,15.

iv. Foi indicado o fator de manutenção de cada etapa do ciclo de vida para cada caso de uso envolvido (f_{UC_Ei}), conforme explicações adiante e tabelas 4.12 e 4.13.

O caso de uso UC_01–Calcular LRC Automático contém 3 transações de manutenção. Uma parte dos requisitos já estava definida e o caso de uso especificado. Por essa razão foi indicado um fator de manutenção de 80%, para a etapa de requisitos. A arquitetura do sistema já estava definida e implementada e a previsão de ajuste na etapa de análise e projeto era insignificante, o que levou a indicação de 0% para o fator de manutenção dessa etapa. No caso da etapa de codificação e da etapa de teste o fator de manutenção foi de 100%, pois foi necessário codificar e testar todas as mudanças propostas no projeto de manutenção, que apesar de serem simples de especificar exigiriam certa complexidade para codificar e testar.

Será mostrada a forma de cálculo da etapa de requisitos do UC_01. As demais etapas desse caso de uso e as etapas do caso de uso UC_28 seguem a mesma forma de cálculo.

A produtividade utilizada na etapa de requisitos foi estimada em 18h.h., por acreditar-se que o analista alocado apresentava um alto grau de conhecimento e uma elevada experiência. Além disso, os requisitos estavam bem estáveis. Assim, utilizando-se a *Eq.* 4.16 para calcular o esforço inicial da etapa de requisitos do UC-01:

$$Esforço_{UC\_E} = TUCP - M_{UC\_E} * EF * \Pr od_E$$

$Esforço_{UC\_01\_Req} = 1{,}15 * 0{,}725 * 18 \Rightarrow Esforço_{UC\_01\_Req} = 15{,}02$ h.h.

Aplicando-se o fator de manutenção de 80% para esta etapa do caso de uso UC_01, tem-se o novo esforço (esforço estimado final) do caso de uso nesta etapa:

$Esforço'_{UC\_01\_Req} = 15{,}02$ h.h. * 0.8 => $Esforço'_{UC\_01\_Req} = 12{,}01$ h.h.

Aplicando-se a Equação 4.19 calcula-se o esforço estimado para o caso de uso UC_01:

$$Esforço_{UC} = \sum_{i=1}^{n} Esforço_{UC\_Ei}$$

$Esforço_{UC\_01} = 15{,}02 + 18{,}78 + 30{,}04 + 11{,}26 \Rightarrow Esforço_{UC\_01} = 75{,}10$ h.h.

A Tabela 4.13 apresenta os valores do tamanho (TUCP-M), esforço estimado inicial (esforço calculado sem a aplicação do fator de manutenção), e o esforço' estimado final (esforço estimado com a aplicação do fator de manutenção), para cada etapa do ciclo de vida do caso de uso UC-01.

Na Tabela 4.13 o valor do Fator-M para o caso de uso UC_01 é calculado da seguinte forma:

$$Fator - M_{UC} = \left(\sum_{i=1}^{n} \left(Fator - M_{UC\_Ei} * Esforço_{UC\_Ei} \right) \right) / Esforço_{UC}$$

$Fator\text{-}M_{UC\_01} = ((1 * 11{,}26) + (1 * 30{,}04) + (0{,}8 * 15{,}02)) / 75{,}10$
$Fator\text{-}M_{UC\_01} = 53{,}31 / 75{,}10$
$Fator\text{-}M_{UC\_01} = 71\%$

Tabela 4.13 – Cálculo do esforço do UC_01 – Projeto Y

Caso de Uso	TUCP-M	Esforço Estimado Inicial	*Fator-M* (f_{UC_Ei})	Esforço' Estimado final	Esforço Real	ERS (erro)
UC_01	**5,76**	**75,10 h.h**	**71%**	**53,31 h.h**	**62,00 h.h**	**16,30%**
Requisito	1,15	15,02 h.h	80%	12,01 h.h	15,00 h.h	24,90%
Análise e Projeto	1,44	18,78 h.h	0%	0,00 h.h	0,00 h.h	0,00%
Codificação	2,30	30,04 h.h	100%	30,04 h.h	33,00 h.h	9,85%
Teste	0,86	11,26 h.h	100%	11,26 h.h	14,00 h.h	24,33%

Além desses valores, a Tabela 4.13 apresenta o esforço real apurado após a conclusão do projeto de manutenção e respectivos erros relativos simétricos (ERS), calculados conforme *Eq.* 4.29. Por exemplo, para o total do UC_01:

ERS = (*Real - Estimado*) / *Estimado*, se Real ≥ Estimado (*Eq.* 4.29)

ERS = (62 -53,31) / 53,31

ERS = 16,30%

O caso de uso UC_28 – Disponibilizar Portfólio não existia anteriormente e foi necessária sua implementação nesse projeto de manutenção. Por isso foi utilizado o Fator-M de 100% para as etapas de requisitos, codificação e teste. Para a etapa de Análise e Projeto foi utilizado um Fator-M de 80%, pois a arquitetura, diagramas e projeto já estavam definidos, bastando realizar algumas alterações. A Tabela 4.14 mostra esses valores:

Tabela 4.14 – Cálculo do esforço do UC_28 – Projeto Y

Caso de Uso	TUCP-M	Esforço estimado Inicial	*Fator-M* (f_{UC_Ei})	Esforço' estimado final	Esforço Real	ERS (erro)
UC_28	**11,51**	**150,22 h.h**	**95%**	**142,70 h.h**	**156,00 h.h**	**9,32%**
Requisito	2,30	30,04 h.h	100%	30,04 h.h	31,00 h.h	3,20%
Análise e Projeto	2,88	37,56 h.h	80%	30,04 h.h	35,00 h.h	16,51%
Codificação	4,61	60,09 h.h	100%	60,09 h.h	64,00 h.h	6,51%
Teste	1,73	22,53 h.h	100%	22,53 h.h	26,00 h.h	15,40%

Conforme a *Eq*. 4.14, o esforço estimado inicial do Projeto Y é:

$$Esforço = TUCP - M * EF * \text{Pr}od$$

$Esforço_{Proj\ Y}$ = 17,27 * 0,725 * 18 => $Esforço_{Proj\ Y}$ = 225,32 h.h

O fator de manutenção total para o projeto é calculado da seguinte maneira:

$$Fator - M_{\text{Pr}oj} = \left(\sum_{i=1}^{n} \left(Fator - M_{Ei} * Esforço_{Ei} \right) \right) / Esforço_{\text{Pr}ojY}$$

$Fator\text{-}M_{UC\_01}$ = ((0,93* 45,06) + (0,53 * 56,34) + (1 * 90,13) + (1*33,79)) / 225,32
$Fator\text{-}M_{UC\_01}$ = 87%

Assim, apresenta-se na Tabela 4.15 o esforço estimado para o Projeto Y com produtividade 18:

Tabela 4.15 – Estimativa de esforço do Projeto Y com produtividade igual a 18

Projeto Y	TUCP-M	Esforço estimado Inicial	*Fator-M* (f_{UC_Ei})	Esforço' estimado final	Esforço Real	ERS (erro)
Etapas	**17,27**	**225,32 h.h**	**87%**	**196,01 h.h**	**218,00 h.h**	**11,22%**
Requisito	3,45	45,06 h.h	93%	42,05 h.h	46,00 h.h	9,39%
Análise e Projeto	4,32	56,34 h.h	53%	30,04 h.h	35,00 h.h	16,51%
Codificação	6,91	90,13 h.h	100%	90,13 h.h	97,00 h.h	7,62%
Teste	2,59	33,79 h.h	100%	33,79 h.h	40,00 h.h	18,38%

O erro relativo simétrico (ERS) do projeto é calculado com base na *Eq.* 4.29:

ERS = (*Real - Estimado*) / *Estimado*, se Real ≥ Estimado

ERS = (218,00 – 196,01) / 196,01

ERS = 11,22%

A estimativa inicial para o esforço de manutenção do projeto Y foi de 225,32 h.h. Após a aplicação do fator de manutenção adequado, o esforço foi estimado em 196,01 h.h. Comparando-se com o esforço real do projeto, pode-se verificar que o erro relativo simétrico (ERS) para esse projeto foi de 11,22%, isto é, o projeto foi subestimado em aproximadamente 11%. Os resultados mostrados utilizaram uma produtividade igual a 18, que foi a produtividade atribuída para as etapas do projeto.

Utilizando-se a produtividade calculada por meio da *Eq.* 4.11, ou seja, produtividade igual a 20, obtém-se os resultados apresentados nas Tabelas 4.16 e 4.17 para os casos de uso afetados:

Tabela 4.16 – Esforço do UC_01 com produtividade 20 – Projeto Y

Caso de Uso	TUCP-M	Esforço estimado Inicial	*Fator-M* (f_{UC_Ei})	Esforço' estimado final	Esforço Real	ERS (erro)
UC_01	**5,76**	**83,45 h.h**	**71%**	**59,25 h.h**	**62,00 h.h**	**4,64%**
Requisito	1,15	16,69 h.h	80%	13,35 h.h	15,00 h.h	12,36%
Análise e Projeto	1,44	20,86 h.h	0%	0,00 h.h	0,00 h.h	0,00%
Codificação	2,30	33,38 h.h	100%	33,38 h.h	33,00 h.h	-1,15%
Teste	0,86	12,52 h.h	100%	12,52 h.h	14,00 h.h	11,82%

Tabela 4.17 – Esforço do UC_28 com produtividade 20 – Projeto Y

Caso de Uso	TUCP-M	Esforço estimado Inicial	*Fator-M* (f_{UC_Ei})	Esforço' estimado final	Esforço Real	ERS (erro)
UC_28	**11,51**	**166,92 h.h**	**95%**	**158,57 h.h**	**156,00 h.h**	**-1,65%**
Requisito	2,30	33,38 h.h	100%	33,38 h.h	31,00 h.h	-7,68%
Análise e Projeto	2,88	41,73 h.h	80%	33,38 h.h	35,00 h.h	4,85%
Codificação	4,61	66,77 h.h	100%	66,77 h.h	64,00 h.h	-4,83%
Teste	1,73	25,04 h.h	100%	25,04 h.h	26,00 h.h	3,83%

A Tabela 4.18 mostra os resultados obtidos para o Projeto Y com produtividade igual a 20.

Tabela 4.18 – Esforço do Projeto Y com produtividade igual a 20

Projeto Y	TUCP-M	Esforço estimado Inicial	*Fator-M* (f_{UC_Ei})	Esforço' estimado final	Esforço Real	ERS (erro)
Etapas	**17,27**	**250,37 h.h**	**87%**	**217,82 h.h**	**218,00 h.h**	**0,08%**
Requisito	3,45	50,07 h.h	93%	46,73 h.h	46,00 h.h	-1,59%
Análise e Projeto	4,32	62,59 h.h	53%	33,38 h.h	35,00 h.h	4,85%
Codificação	6,91	100,15 h.h	100%	100,15 h.h	97,00 h.h	-3,25%
Teste	2,59	37,56 h.h	100%	37,56 h.h	40,00 h.h	6,50%

Percebe-se que o erro relativo simétrico (ERS) foi praticamente nulo, apontando um resultado muito satisfatório. Presume-se que este resultado deveu-se em parte pela diminuição da subjetividade na atribuição da produtividade, quando a *Eq.* 4.11 é utilizada.

4.4.9 Estimativa de Prazo

Para calcular a estimativa de prazo de cada caso de uso foi utilizada a alocação de um profissional em cada etapa do ciclo de vida, trabalhando 8 horas por dia. Assim, as Tabelas 4.19 e 4.20 apresentam respectivamente o prazo necessário para os casos de uso UC_01 e UC_28, com produtividade igual a 18.

Tabela 4.19 – Prazo do UC_01 – Projeto Y

Etapa	Esforço estimado final	Prazo Estimado	Prazo Real	ERS (erro)
Requisito	12,01 h.h	1,50 dias	1,88 dias	24,90%
Análise e Projeto	0,00 h.h	0,00 dias	0,00 dias	0,00%
Codificação	30,04 h.h	3,76 dias	4,13 dias	9,85%
Teste	11,26 h.h	1,41 dias	1,75 dias	24,33%
Total	**53,31 h.h**	**6,66 dias**	**7,75 dias**	**16,30%**

Evidentemente, o prazo apresentado nesta seção ainda deve ser transportado para ferramenta de construção de cronograma do projeto, pois deve-se levar em consideração a possibilidade da execução de atividades em paralelo.

Tabela 4.20 –Prazo do UC_28 – Projeto Y

Etapa	Esforço estimado final	Prazo Estimado	Prazo Real	ERS (erro)
Requisito	30,04 h.h	3,76 dias	3,88 dias	3,20%
Análise e Projeto	30,04 h.h	3,76 dias	4,38 dias	16,51%
Codificação	60,09 h.h	7,51 dias	8,00 dias	6,51%
Teste	22,53 h.h	2,82 dias	3,25 dias	15,40%
Total	**142,70 h.h**	**17,84 dias**	**19,50 dias**	**9,32%**

A Tabela 4.21 apresenta a estimativa do prazo necessário para o Projeto Y, com produtividade 18:

Tabela 4.21 – Prazo do Projeto Y

Etapa	Esforço estimado final	Prazo Estimado	Prazo Real	ERS (erro)
Requisito	42,05 h.h	5,26 dias	5,75 dias	9,39%
Análise e Projeto	30,04 h.h	3,76 dias	4,38 dias	16,51%
Codificação	90,13 h.h	11,27 dias	12,13 dias	7,62%
Teste	33,79 h.h	4,22 dias	5,00 dias	18,38%
Total	**196,01 h.h**	**24,50 dias**	**27,25 dias**	**11,22%**

4.4.10 Estimativa de Custo

O custo do projeto pode ser determinado utilizando-se as equações 4.26 e 4.27. Na organização onde o projeto foi realizado costuma-se estimar o custo apenas dos projetos que serão licitados. Como o projeto utilizado não foi alvo de licitação, serão utilizados aqui valores hipotéticos para cada papel exercido nas etapas do ciclo de vida, objetivando exemplificar a estimativa de custo do projeto.

Assim, será estabelecido que na etapa de requisitos o principal papel é do analista de sistema, na etapa de Análise e Projeto o principal papel é do arquiteto, na etapa de codificação o principal papel é do codificador e na etapa de teste o principal papel é do analista de teste. Supondo ainda que o valor de cada papel exercido seja de R$ 40,00 para analista de sistemas, R$ 50,00 para arquiteto, R$ 30,00 para codificador e R$ 40,00 para analista de teste, tem-se:

$$\text{Custo}_{Ei(\text{unidade\_monetária})} = \text{Esforço}_{Ei} * \text{Valor}_{(\text{papel})}$$

$\text{Custo}_{Req} =$ 42,05 h.h * R$ 40,00 => $\text{Custo}_{Req} =$ R$ 1.682,00

$\text{Custo}_{A\&P} =$ 30,04 h.h * R$ 45,00 => $\text{Custo}_{A\&P} =$ R$ 1.351,80

$\text{Custo}_{Cod} =$ 90,13 h.h * R$ 30,00 => $\text{Custo}_{Cod} =$ R$ 2.703,90

$\text{Custo}_{Tst} =$ 33,79 h.h * R$ 40,00 => $\text{Custo}_{Tst} =$ R$ 1.351,60

Para a estimativa do custo total do projeto:

$$Custo_{(unidade\_monetária)} = \sum_{i=1}^{n} Custo_{Ei}$$

Custo = R$ 1.682,00 + R$ 1.351,80 + R$ 2.703,90 + R$ 1.351,60

Custo = R$ 7.089,30

4.5 Comparação entre TUCP e TUCP-M

O Projeto Y também foi estimado utilizando-se a TUCP, objetivando comparar os resultados apresentados com a utilização da TUCP-M. Comparou-se as duas técnicas com o valor de produtividade igual a 18 e também com produtividade igual a 20.

Utilizar a TUCP significa:

- Calcular o esforço de um projeto de manutenção baseado em uma técnica mais direcionada a projetos de desenvolvimento de software.
- Levar em consideração todas as transações dos casos de uso afetados e não somente as incluídas, alteradas e excluídas.
- Não utilizar o fator de manutenção (f_{UC_Ei}), para calibrar o esforço por etapa do ciclo de vida, ou utilizar sempre igual a 100%.

As estimativas calculadas são apresentadas nas Tabelas 4.22, 4.23 e 4.24:

Tabela 4.22 – Estimativa de esforço do UC_01 utilizando a TUCP – Projeto Y

Etapa	Esforço estimado TUCP com prod=18	Esforço estimado TUCP com prod=20	Esforço Real	ERS TUCP com prod=18	ERS TUCP-M com Prod=18	ERS TUCP com prod=20	ERS TUCP-M com Prod=20
UC_01	**138,27 h.h**	**153,64 h.h**	**62,00 h.h**	**-123,02%**	**16,30%**	**-147,81%**	**4,64%**
Requisito	27,65 h.h	30,73 h.h	15,00 h.h	-84,33%	24,90%	-104,87%	12,36%
Análise e Projeto	34,57 h.h	38,41 h.h	0,00 h.h	*******	0,00%	*******	0,00%
Codificação	55,31 h.h	61,46 h.h	33,00 h.h	-67,61%	9,85%	-86,24%	-1,15%
Teste	20,74 h.h	23,04 h.h	14,00 h.h	-48,14%	24,33%	-64,57%	11,82%

As estimativas calculadas pela TUCP para o UC_01 foram muito acima do realizado, principalmente em virtude de:

- A quantidade de transações levada em consideração foi de 5 ao invés das 3 transações contadas na TUCP-M, fato que elevou a complexidade do caso de uso de simples para média, implicando no aumento do tamanho calculado.
- A não utilização do fator de manutenção (f_{UC_Ei}), para calibrar o esforço por etapa do ciclo de vida, elevou o esforço estimado como se tivesse utilizado f_{UC_Ei}= 100% em todas as etapas.

No caso do UC_28 a diferença entre o estimado e o realizado foi menor porque o UC_28 tem comportamento similar a um projeto de desenvolvimento por ser um caso de uso totalmente novo. A quantidade de transações foi a mesma e o f_{UC_Ei}=100%, em todas as etapas, exceto para a etapa de Análise e Projeto que recebeu fator de manutenção igual a 80%.

O total estimado pela TUCP foi subestimado e pela TUCP-M superestimado. Percebe-se que com a produtividade igual a 20, a TUCP obteve uma estimativa levemente mais aprimorada. Nesse caso, talvez o percentual de manutenção da etapa de requisitos tenha sido calibrado com um valor um pouco acima do necessário. Entretanto, com a produtividade igual a 18, que foi a utilizada no Projeto Y, a TUCP-M foi mais acurada. A Tabela 4.23 apresenta as estimativas para o UC_28 utilizando a TUCP.

Tabela 4.23 – Estimativa de esforço do UC_28 utilizando a TUCP – Projeto Y

Etapa	Esforço estimado TUCP com prod=18	Esforço estimado TUCP com prod=20	Esforço Real	ERS TUCP com prod=18	ERS TUCP-M com Prod=18	ERS TUCP com prod=20	ERS TUCP-M com Prod=20
UC_28	**138,27 h.h**	**153,64 h.h**	**156,00 h.h**	**12,82%**	**9,32%**	**1,54%**	**-1,65%**
Requisito	27,65 h.h	30,73 h.h	31,00 h.h	12,12%	3,20%	0,88%	-7,68%
Análise e Projeto	34,57 h.h	38,41 h.h	35,00 h.h	1,24%	16,51%	-9,74%	4,85%
Codificação	55,31 h.h	61,46 h.h	64,00 h.h	15,71%	6,51%	4,13%	-4,33%
Teste	20,74 h.h	23,04 h.h	26,00 h.h	25,36%	15,40%	12,85%	3,83%

A Tabela 4.24 apresenta os valores das estimativas calculadas para o Projeto Y.

Tabela 4.24 – Estimativa de esforço do Projeto Y utilizando a TUCP e comparação dos resultados apresentados pelas TUCP e TUCP-M

Etapa	Esforço estimado TUCP com prod=18	Esforço estimado TUCP com prod=20	Esforço Real	ERS TUCP com prod=18	ERS TUCP-M com Prod=18	ERS TUCP com prod=20	ERS TUCP-M com Prod=20
Etapas	**276,54 h.h**	**307,28 h.h**	**218,00 h.h**	**-26,85%**	**11,22%**	**-40,95 %**	**0,08%**
Requisito	55,30 h.h	61,46 h.h	46,00 h.h	-20,22%	9,39%	-33,61 %	-1,59%
Análise e Projeto	69,14 h.h	76,82 h.h	35,00 h.h	-97,54%	16,51%	-119,42 %	4,85%
Codificação	110,62 h.h	122,92 h.h	97,00 h.h	-14,04%	7,62%	-26,72 %	-3,25%
Teste	41,48 h.h	46,08 h.h	40,00 h.h	-3,70%	18,38%	-15,20 %	6,50%

Verificou-se que com produtividade de 18, o esforço calculado pela TUCP foi superestimado em 26,85% do esforço real e o esforço calculado pela TUCP-M foi subestimado em 11,22%. Com a produtividade igual a 20, o esforço calculado pela TUCP foi superestimado em 40,95% do esforço real e o esforço calculado pela TUCP-M foi subestimado em 0,08%. Portanto, em ambos os casos a TUCP-M foi bem mais aprimorada no cálculo das estimativas para o Projeto Y.

4.6 Conclusão

A engenharia de software como uma disciplina da engenharia teve um significativo crescimento. Entretanto, sua maturidade não foi proporcional a este crescimento. Uma das habilidades mais importante, que ainda necessita amadurecer mais é a habilidade de estimar o tamanho e conseqüentemente o esforço, em um grau razoável de exatidão. Apesar do desenvolvimento de uma quantidade significativa de técnicas para estimar o esforço, a maioria é mais direcionada ao desenvolvimento do que à manutenção de software (BHATT *et al*, 2004).

Este capítulo apresentou a proposta da técnica TUCP-M para estimar projetos de manutenção de software baseada em casos de uso. Acredita-se que a técnica venha a auxiliar fortemente o cálculo de estimativas mais acuradas para projetos dessa natureza. A contagem apenas das transações que foram incluídas, alteradas ou excluídas e a utilização do fator de manutenção por etapa do ciclo de vida do projeto, são fatores que auxiliam na acurácia da técnica.

Verificou-se também que as estimativas calculadas para o projeto de manutenção de software mostrado como exemplo foram mais próximas do realizado quando utilizada a TUCP-M ao invés da TUCP, mais direcionada a projetos de desenvolvimento de software.

No Capítulo 5 são apresentados alguns projetos de manutenção de software que foram estimados com a TUCP-M e a comparação dos resultados apresentados com o real esforço demandado para tais projetos. Além dos exemplos, a utilização da técnica também proporcionou na instituição que utilizou a TUCP-M, a visualização de algumas necessidades na organização e no processo de desenvolvimento e manutenção utilizados. Esses fatos também são mencionados no Capítulo 5.

Capítulo 5
Estudo de Casos

*Este capítulo apresenta o resultado da aplicação da **TUCP-M** para estimar projetos de manutenção de software, realizados em uma Instituição federal de grande porte, objetivando avaliar a utilização da técnica proposta.*

Durante a execução de diversos projetos de software em uma empresa pública de grande porte, que participa do sistema financeiro do país, pôde-se perceber o que as pesquisas já apontavam: a maioria dos projetos de software era referente à manutenção.

No caso de projetos de desenvolvimento de software, era utilizada a TUCP. Porém, para os projetos de manutenção de software carecia-se ainda de uma técnica que calculasse suas estimativas de forma mais simples e precisa e que pudesse ser utilizada logo nas etapas iniciais do projeto. Neste contexto, foi proposta a utilização da TUCP-M (Pontos de caso de Uso Técnico para Manutenção de Software) para estimar projetos dessa natureza.

Os projetos de manutenção de software dessa organização, cujos requisitos dos sistemas são descritos por meio de casos de uso, estão utilizando desde então a TUCP-M para calcular suas estimativas de tamanho, esforço e prazo. Em virtude de ser uma organização do governo federal, o custo geralmente é calculado quando é necessária a realização de processo licitatório para a execução do projeto.

Na Seção 5.1 descreve-se o perfil da organização onde foram realizados os experimentos. Na Seção 5.2 discorre-se resumidamente sobre a metodologia utilizada na execução dos projetos de manutenção de software. As Seções 5.3, 5.4, 5.5 e 5.6 descrevem os projetos selecionados como exemplo. A Seção 5.7 comenta algumas situações verificadas durante a aplicação da TUCP-M e as soluções adotadas quando necessário. Por último, as conclusões desse capítulo são mencionadas na Seção 5.8.

5.1 Perfil da Organização

A organização escolhida para a aplicação da TUCP-M em seus projetos de manutenção de software participa do sistema financeiro nacional brasileiro, é do governo federal e tem 56 (cinqüenta e seis) anos de existência.

Sua sede é na cidade de Fortaleza, Estado do Ceará. Atua nos nove estados do Nordeste, no norte de Minas Gerais e o norte do Espírito Santo, exercendo trabalho de atração

de investimentos, estrutura do desenvolvimento por meio de projetos de grande impacto, além de ser um agente de intermediação financeira. Atua também na área de microcrédito.

Os sistemas dessa organização atendem aos órgãos de sua Direção Geral e a uma rede de 190 agências espalhadas por sua área de atuação.

A Direção Geral está dividida em departamentos denominados ambientes. Dentre estes, existe o Ambiente de Sistemas de Informação, responsável, dentre outros, pelo desenvolvimento e manutenção dos sistemas da organização.

Atualmente, 301 sistemas estão ativos, isto é, são passíveis de receber solicitação de mudanças ou evolução. Esses sistemas dão apoio aos processos internos da organização e sustentam os negócios realizados com seus clientes externos. No ano de 2007, foram realizados 179 projetos de software nessa organização, divididos em desenvolvimento e manutenção, afetando 166 dos seus sistemas.

Nessa organização, considera-se como um projeto de software o desenvolvimento de um novo aplicativo ou o atendimento a determinada quantidade de demandas de manutenção referente a um sistema já existente. O objetivo desses projetos é gerar uma primeira versão de um novo sistema ou gerar uma nova versão de um sistema existente.

O Ambiente de Sistemas de Informação atende a 43 Ambientes gestores dos sistemas existentes. Está divido em células, cada uma delas responsável por determinadas atividades do processo de desenvolvimento e manutenção de software da organização, conforme Figura 5.1. A equipe do Ambiente de Sistemas de Informação é composta de 288 colaboradores, dos quais 149 são funcionários e 139 terceirizados.

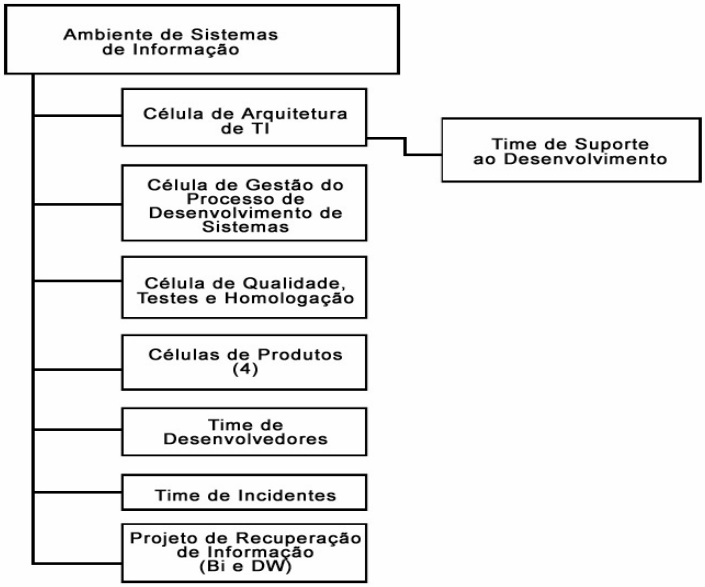


Figura 5.1 – Estrutura do Ambiente de Sistemas de Informação da Organização

Cada uma das 4 células de produto mencionadas na Figura 5.1 é responsável por um conjunto de sistemas, que atendem a um determinado grupo de Ambientes gestores desses aplicativos, os quais atuam em diferentes segmentos dentro da organização. Dessa forma, cada célula de produto é responsável pelo recebimento e atendimento das demandas dos usuários gestores dos sistemas sob sua responsabilidade. Entende-se por demandas, as requisições dos usuários gestores, que irão implicar em alteração de funcionalidades nos sistemas existentes ou criação de novos sistemas. Em 2007, foi atendido um total de 4.982 demandas.

As células de produto são compostas por equipes de analistas de sistemas responsáveis pelos aplicativos de cada segmento. A célula de arquitetura é responsável pela definição da arquitetura do projeto e pelo suporte às equipes responsáveis pelos projetos de software, nos padrões e ferramentas referentes à arquitetura definida. A célula de Qualidade Teste e Homologação é responsável por especificar e preparar dados de testes, planejar, projetar e executar os testes funcionais integrados de sistemas, dentre outras atividades. A célula de Gestão do Processo responde pela definição, implantação, monitoração e evolução contínua do processo de desenvolvimento de software e demais processos derivados. A célula de teste está em fase de estruturação e ainda não tem condições de atender a todos os projetos, razão pela qual os testes de alguns projetos são realizados apenas pelos analistas de sistemas da equipe responsável pelo projeto, antes de enviar para o aceite do usuário gestor. O time de desenvolvedores é totalmente terceirizado e os programadores são alocados aos projetos por demanda, à medida que as especificações estão completas e prontas para serem enviadas para implementação.

Desde 2005, essa organização possui um processo de desenvolvimento e manutenção de software baseado no RUP – *Rational Unified Process* (2003), que prevê as seguintes fases para seus projetos de software:

- Iniciação: cujo principal objetivo é identificar o escopo do sistema
- Elaboração: objetiva obter arquitetura estável para o sistema
- Construção: fase para desenvolver versão operacional do sistema
- Transição: realizar a implantação da versão final do sistema

As seguintes disciplinas com seus respectivos artefatos estão previstos na metodologia adotada:

- Disciplina Requisitos: Documento de Visão, Plano de Gerenciamento de Requisitos, Glossário, Regras de Negócio, Diagrama de Casos de Uso, Especificação de Casos de Uso, Especificação de Requisitos, Especificação Suplementar.
- Disciplina Análise e Projeto: Prova de Conceito Arquitetural, Documento de Arquitetura, Modelo de Análise, Modelo de *Design*, Modelo de Dados, Modelo de Implantação, Realização de Casos de Uso, Protótipo de Interface do Usuário, Lista de Materiais, Especificação Suplementar.
- Disciplina Implementação: *Help Online*, Componentes, *Builds*, Código Fonte, Classes de Teste, Resultados dos Testes Unitários, Relatório de Cobertura dos Testes Unitários, Plano de Implantação e Notas de Liberação, Lista de Materiais.
- Disciplina de Testes: Plano de Testes, Planilha de Cenários e Casos de Testes, *Script* de Testes Manuais, *Script* de Testes Automatizados, *Log* de Resultados dos Testes, Dados de Testes Identificados, Defeitos Encontrados, Sumário de Avaliação de Teste.
- Disciplina de Implantação: Plano de Implantação, Artefatos de Instalação, Material de Treinamento, Material de Suporte para Usuário.
- Disciplina de Gerenciamento de Projetos: Termo de Abertura do Projeto, Planilha de Estimativa do Projeto, Plano de Desenvolvimento de Software, Plano de Iteração, Plano de Gerenciamento de Riscos, Lista de Riscos, Avaliação da Iteração, Termo de Aceite.
- Disciplina de Gerenciamento de Configuração e Mudanças: Plano de gerenciamento de configuração, Repositório do projeto.
- Disciplina de Ambiente: Infra-estrutura de desenvolvimento.

As principais tecnologias e linguagens adotadas na Instituição são:

- Mainframe: Enterprise Cobol, Mantis
- Cliente Servidor: Visual Basic, PowerBuilder
- WEB: HTML, ASP, ASP.NET, C#, Java

Para cada projeto iniciado, que pode ser de desenvolvimento de novo aplicativo ou manutenção em um sistema já existente, é determinada uma equipe responsável. A equipe deverá ser constituída de gerente de projeto, analista de sistema, arquiteto, desenvolvedores e

analista de teste, em quantidades que dependem do porte e da criticidade do projeto, mas também fortemente influenciada pela quantidade de recursos disponíveis.

A organização possui ferramenta para o cadastramento das demandas pelo usuário gestor e gerenciamento destas demandas. Cada demanda cadastrada é direcionada para a célula de produto responsável. Para os sistemas já existentes e que sofrem constantes manutenções, costuma-se agrupar um conjunto de demandas cadastradas, dando início a um projeto de manutenção, que será planejado e executado conforme as fases previstas na metodologia adotada.

5.2 Metodologia

Na organização onde a TUCP-M está sendo utilizada, dentre as funções que estão sob a responsabilidade das células de produtos, pode-se destacar:

- Gerenciar o atendimento às demandas de desenvolvimento e evolução de sistemas
- Gerenciar projetos de desenvolvimento e evolução de software
- Planejar fases, iterações, treinamento e implantação de sistemas.

Para realizar estas atividades é inegável a necessidade do conhecimento do tamanho, esforço, prazo e custo, tanto de projetos de desenvolvimento como de projetos de manutenção de software. A primeira estimativa dos projetos é calculada pelos analistas dos sistemas ou gerentes de projetos responsáveis, ao final da fase de Iniciação.

Para esse trabalho, foi desenvolvida uma ferramenta em planilha eletrônica, que automatizou a TUCP e a TUCP-M e facilitou e agilizou os cálculos das estimativas de projeto de software, permitindo a coleta dos valores efetivamente realizados nesses projetos e a comparação entre os valores estimados e os realizados.

De uma maneira geral, o escopo de um projeto de manutenção é determinado e estimado da seguinte forma:

- Para a maioria dos sistemas que sofrem manutenção na organização, já existe um analista e gerente de projeto dedicado para atender suas demandas de manutenção. Pelo simples motivo de que geralmente estes sistemas são os mais críticos da organização.
- De posse das demandas cadastradas é realizada reunião com o usuário gestor, objetivando priorizar um conjunto de demandas a serem atendidas, que formam a partir dessa priorização, um projeto de manutenção.

- O gerente de projeto participa de mais de um projeto de manutenção ou desenvolvimento ao mesmo tempo. Entretanto, os analistas de sistemas costumam participar de um único projeto por vez. Dependendo do porte e da criticidade do projeto, mais de um analista de sistema pode ser alocado ao mesmo projeto.
- Gerente de projeto e analista de sistema são responsáveis pela fase de iniciação, quando são identificados os requisitos afetados pelo projeto de manutenção.
- Ao final da fase de Iniciação o documento de visão do projeto é fechado e são identificados os casos de uso afetados com a manutenção.
- Como nas demais organizações, é necessário calcular as estimativas do projeto com a maior brevidade possível. Assim, de posse da relação dos casos de uso afetados, a equipe é capaz de construir um esboço dos casos de uso e, com base neste esboço, estimar a quantidade de transações incluídas, alteradas ou excluídas em virtude das demandas a serem atendidas.
- Os atores relacionados aos casos de uso afetados com a manutenção são identificados.
- De posse da quantidade de transações e das demais informações necessárias ao cálculo das estimativas relacionadas no Capítulo 4, estes dados são transportados para a planilha TUCP-M.
- Nesse momento, o ideal é conhecer a quantidade e o perfil das pessoas que comporão a equipe, pois isso influencia fortemente o cálculo da estimativa de esforço e a construção da primeira versão do cronograma do projeto. Muitas vezes isso não é possível, o que pode levar a percentuais de erro acima do limite aceitável.
- A quantidade e a experiência dos desenvolvedores podem variar dependendo da disponibilidade destes recursos na época da realização da etapa de codificação. Um planejamento de alocação é construído, mas podem ocorrer algumas mudanças durante a efetiva execução do projeto.
- A planilha calcula o esforço necessário para cada caso de uso e dependendo dos recursos disponíveis para o projeto e das dependências entre as atividades que serão realizadas, o cronograma do projeto é construído. O plano do projeto, o planejamento de suas iterações e entregas é construído com base nas estimativas calculadas.

Está previsto na metodologia de desenvolvimento de software utilizada na organização, o refinamento das estimativas durante o decorrer do projeto. Algumas ocorrências podem afetar significativamente a estimativa feita ao final da fase de Iniciação.

As principais são: mudança no escopo e mudança na equipe prevista para o projeto. Nestes casos, os solicitantes devem ser comunicados do impacto gerado. Assim, ou negocia-se o planejamento das entregas com seu conjunto de requisitos a serem atendidos ou o cronograma do projeto é refeito à luz das mudanças surgidas.

Os projetos apresentados nas Seções 5.3 a 5.6 são identificados respectivamente como Projeto A, Projeto B, Projeto C, Projeto D. Para esses projetos os percentuais de esforço por etapa do ciclo de vida utilizados estão apresentados na Tabela 5.1. Estes valores foram definidos com base nos valores utilizados em Monteiro (2005), Vazquez *et al* (2007) e na experiência da organização.

Esses percentuais e as próprias etapas do ciclo de vida dos projetos de software da organização sofreram adaptações, com o intuito de corrigir alguns problemas de experiência da equipe e esforço necessário por etapa do ciclo de vida.

Tabela 5.1 – Percentual de esforço por etapa do clico de vida

Etapa do Ciclo de Vida	Percentual de Esforço
Requisito e Análise - (*R&A*)	27%
Projeto – (*Pro*)	18%
Codificação – (*Cod*)	40%
Teste – (*Tst*)	15%

Foi o uso da TUCP-M que possibilitou a identificação da necessidade dessas adaptações, conforme comentado durante a apresentação dos projetos utilizados como exemplos nas Seções 5.3 a 5.6 e na Seção 5.7.

5.3 Projeto A

O Projeto A de manutenção foi realizado em um sistema de grande porte, desenvolvido na plataforma Cliente/Servidor, duas camadas, utilizando a linguagem Powerbuilder. O sistema trata da administração das operações de crédito contratadas com os clientes da organização.

Como o sistema é um dos mais críticos da organização e um dos que mais recebe solicitações de mudança, conta com uma equipe de 8 analistas e 1 gerente de projetos. Entretanto, é comum existirem em média de 4 a 5 projetos em execução ao mesmo tempo. Por essa razão, cada projeto conta com uma pequena quantidade de analistas e o gerente de projeto é compartilhado entre esses projetos.

O Projeto A de manutenção contou com 1 analista com conhecimento e experiência médios no sistema, pois se tratava de funcionário contratado há pouco tempo na organização e

1 programador com pouca experiência no sistema e na linguagem utilizada. Nesta época, a empresa terceirizada responsável pelos desenvolvedores estava com problemas para contratar no mercado profissional com experiência em PowerBuilder.

Como o gerente de projeto estava responsável por mais outros quatro projetos, o analista de sistemas ficou responsável pelo cálculo das estimativas.

O projeto foi aberto com apenas uma demanda, por tratar-se de demanda com uma complexidade razoável. Seu objetivo era viabilizar o desembolso dos valores relacionados às operações de crédito por meio de DOC (Documento de Ordem de Crédito) em contas previamente determinadas por seus clientes.

Depois de realizada a análise de impacto, quando foram identificados os casos de uso, atores e transações de manutenção, o cálculo da estimativa de tamanho do Projeto A foi realizado seguindo as seguintes etapas:

- Contagem dos atores - (UAW)
- Contagem dos casos de uso - (TUUCW)
- Cálculo dos pontos de casos de uso não ajustados - (TUUCP)
- Cálculo dos fatores de complexidade técnica - (TCF)
- Cálculo da estimativa de tamanho da manutenção - (TUCP-M)

Contagem dos Atores - (UAW)

Neste projeto, 6 (seis) atores foram afetados com a manutenção, 3 (três) de simples complexidade e 3 (três) de média complexidade, com o peso total não ajustado dos atores (*UAW*) de 9 (nove), como mostrado na Tabela 5.2:

Tabela 5.2 – Contagem dos Atores – Projeto A

Qtde. Atores	Complexidade	Peso	Total
3	Simples	1	3*1 = 3
3	Médios	2	3*2 = 6
			UAW = 9

Contagem dos Casos de Uso - (TUUCW)

Fazem parte do Projeto A 6 casos de uso, estimados segundo a TUCP-M. Após a apuração da quantidade de transações requeridas para inclusão (t_i), alteração (t_a), ou exclusão (t_e) de passos do fluxo de eventos, que foram afetados pela manutenção, obteve-se o total de transações de manutenção (t_m) para cada um deles. Assim sendo, a contagem dos casos de uso do Projeto A (TUUCW) foi de 85, conforme Tabela 5.3:

Tabela 5.3 – Contagem dos Casos de Uso – Projeto A

Caso de Uso	Qtde. transações de manutenção	Complexidade	TUUCW
UC_01	11 t_m	Complexo	15
UC_02	12 t_m	*n*-complexo	20
UC_03	11 t_m	Complexo	15
UC_04	07 t_m	Médio	10
UC_05	11 t_m	Complexo	15
UC_06	05 t_m	Médio	10
Total			**85**

Cálculo dos Pontos de Casos de Uso não Ajustados - (TUUCP)

Para o cálculo dos pontos de casos de uso não ajustados utilizou-se a *Eq.* 4.2:

TUUCP = Σ UAW + Σ TUUCW → TUUCP = 9 + 85 → TUUCP = 94

Cálculo dos Fatores de Complexidade Técnica - (TCF)

A Tabela 5.4 apresenta os valores indicados para os fatores de complexidade técnica do Projeto A. Cada nota foi indicada levando em consideração os valores mostrados na Tabela 4.4 de termos lingüísticos.

Tabela 5.4 – Fatores de Complexidade Técnica – Projeto A

Fator	Descrição	Peso	Nota	Produto
T1	Sistemas Distribuídos	2,0	3	6
T2	Desempenho da Aplicação	1,0	4	4
T3	Eficiência do Usuário-Final (*on-line*)	1,0	5	5
T4	Processamento Interno Complexo.	1,0	5	5
T5	Reusabilidade do código em outras aplicações	1,0	0	0
T6	Facilidade de Instalação	0,5	5	2,5
T7	Usabilidade (facilidade operacional)	0,5	5	2,5
T8	Portabilidade	2,0	0	0
T9	Facilidade de Manutenção	1,0	4	4
T10	Concorrência	1,0	1	1
T11	Características especiais de segurança	1,0	3	3
T12	Acesso direto para terceiros	1,0	0	0
T13	Facilidades especiais de treinamento	1,0	2	2
Total		-	-	35

O cálculo do TFator é realizado por: TFator = ΣFator = 35. Utilizando a *Eq.* 4.5:

TCF = 0,6 + (0,01 * TFator) => TCF = 0,95

Cálculo da estimativa de tamanho da manutenção - (TUCP-M)

Utilizando-se a *Eq.* 4.4, calcula-se o tamanho do Projeto A:

TUCP-$M_{Proj\ A}$ = TUUCP * TCF TUCP-$M_{Proj\ A}$ = 94 * 0,95 => TUCP-$M_{Proj\ A}$ = 89,30

Para calcular o tamanho da manutenção de cada caso de uso foi utilizada a *Eq.* 4.8:

$$TUCP - M_{(UC)} = \left(\frac{TUCP - M}{\sum TUUCW} \right) * TUUCW_{(UC)}$$ Onde:

TUCP-$M_{Proj\ A}$ = 89,30

$TUUCW_{UC\_01}$ = 15 $TUUCW_{UC\_02}$ = 20 $TUUCW_{UC\_03}$ = 15

$TUUCW_{UC\_05}$ = 10 $TUUCW_{UC\_06}$ = 15 $TUUCW_{UC\_07}$ = 10

A Tabela 5.5 mostra o tamanho de cada caso de uso por etapa do ciclo de vida.

Tabela 5.5 – Estimativa de tamanho dos Casos de Uso por etapa do ciclo de vida–Projeto A

Caso de Uso	Requisito e Análise	Projeto	Codificação	Teste	Total
UC_01	4,25	2,84	6,30	2,36	15,76
UC_02	5,67	3,78	8,40	3,15	21,01
UC_03	4,25	2,84	6,30	2,36	15,76
UC_04	2,84	1,89	4,20	1,58	10,51
UC_05	4,25	2,84	6,30	2,36	15,76
UC_06	2,84	1,89	4,20	1,58	10,51

Conhecido o tamanho do Projeto A, pode-se estimar seu esforço. Para estimar o esforço do projeto deve-se calcular os fatores de ambiente (EF), a produtividade (Prod) e por fim o esforço de manutenção.

Cálculo dos Fatores de Ambiente - (EF)

A Tabela 5.6 apresenta os fatores de ambiente para o Projeto A. Foram atribuídas as notas referentes a cada fator ambiental conforme Tabela 4.6 de termos lingüísticos. A coluna "produto" indica o valor da multiplicação entre o valor do fator ambiental e seu peso. Para calcular o valor do fator ambiental do Projeto A (EF_{projA}), utilizou-se a *Eq.* 4.10.

Tabela 5.6 – Fatores de Ambiente – Projeto A

Fator	Descrição	Peso	Nota	Produto
F1	Familiaridade com o processo de desenvolvimento	1,5	3	**4,5**
F2	Experiência com a aplicação em desenvolvimento	0,5	3	**1,5**
F3	Experiência com orientação a objeto	1	3	**3**
F4	Capacidade de análise	0,5	4	**2**
F5	Motivação	1	5	**5**
F6	Requisitos estáveis	2	3	**6**
F7	Colaboradores de meio período	-1	5	**-5**
F8	Dificuldade na linguagem de programação	-1	3	**-3**
EFator				**14**

EFator $_{projA}$= 14

EF_{projA} = 1,4 + (-0,03 * 14) => EF_{projA} = 0,98

Cálculo da Produtividade – (Prod)

A produtividade referente a cada etapa do ciclo de vida do projeto foi calculada levando-se em consideração o perfil do profissional mais relevante em cada etapa. No caso do *Projeto A*, a etapa de Requisito e Análise e a etapa de Projeto foram realizadas pelo analista de sistemas. Assim, foi calculada a produtividade dessas etapas da seguinte forma:

- Os fatores ambientais receberam os valores: **F1**=3, **F2**=3, **F3**=3, **F6**=3
- Utilizando a *Eq. 4.11*, onde, *Prod* = 30 – 0,6 *x*:

  $x = x_1 + x_2$

  x_1 = (1,5 *F1* + 0,5*F2* + *F3*) → variável experiência da equipe

  x_1 = (1,5*3) + (0,5*3) + 3

  x_1 = 4,5 + 1,5 + 3 => x_1 = 9

  x_2 = 2 *F6* → variável estabilidade dos requisitos

  x_2 = 2* 3 => x_2 = 6

  $x = x_1 + x_2$ => x = 15

  Prod = 30 – 0,6 x = 21

  Assim, a produtividade para Etapa$_{R\&A}$ = 21 e a produtividade para Etapa$_{Pro}$ = 21.

Para a etapa de codificação os fatores ambientais receberam os valores: **F1**=0, **F2**=0, **F3**=1, **F6**=3, em virtude da baixa experiência do desenvolvedor:

- Utilizando a *Eq. 4.11*, onde, *Prod* = 30 – 0,6 *x*:

  $x = x_1 + x_2$

  x_1 = (1,5 *F1* + 0,5*F2* + *F3*) → variável experiência da equipe

  x_1 = (1,5*0) + (0,5*0) + 1

  x_1 = 0 + 0 + 1 => x_1 = 1

  x_2 = 2 *F6* → variável estabilidade dos requisitos

  x_2 = 2* 3 => x_2 = 6

  $x = x_1 + x_2$ => x = 7

  Prod = 30 – 0,6 x = 26

  Assim, a produtividade para Etapa$_{Cod}$ = 26.

Para a etapa de teste os fatores ambientais receberam os valores: **F1**=2, **F2**= 0, **F3**=1, **F6**=3.

- Utilizando a *Eq. 4.11*, onde, $Prod = 30 - 0{,}6\,x$:

  $x_1 = (1{,}5\,F1 + 0{,}5F2 + F3) \rightarrow$ variável experiência da equipe

  $x_1 = (1{,}5*2) + (0{,}5*0) + 1 \Rightarrow 3 + 0 + 1 \Rightarrow x_1 = 4$

  $x_2 = 2\,F6 \rightarrow$ variável estabilidade dos requisitos

  $x_2 = 2*\,3 \Rightarrow x_2 = 6$

  $x = x_1 + x_2 \Rightarrow x = 10 \quad \Rightarrow \; Prod = 30 - 0{,}6\,x = 24$

  Assim, a produtividade para $\text{Etapa}_{Tst} = 24$.

A Tabela 5.7 apresenta a produtividade para cada uma das etapas do ciclo de vida do Projeto A:

Tabela 5.7 – Produtividade por Etapa do Ciclo de Vida – Projeto A

Etapa	Produtividade
Requisito e Análise	21
Projeto	21
Codificação	26
Teste	24

Cálculo do Esforço de Manutenção – Projeto A

Para o cálculo do esforço de manutenção referente ao Projeto A foram utilizadas as equações apresentadas no Capítulo 4. A Tabela 5.8 apresenta os cálculos realizados para o UC_01. O fator de manutenção de cada etapa considerou um esforço relativo para cada uma delas, entretanto, foi indicado um percentual menor para a etapa de teste, pois os testes foram realizados pelo analista de sistema e não pela equipe de teste. Dessa forma, o esforço de construção do planejamento dos casos de testes foi abolido. Mesmo assim, o esforço para teste foi menor. Talvez 30% seria um fator de manutenção mais adequado para a etapa de teste. Utilizou-se a *Eq*. 4.29 para o cálculo do erro relativo simétrico.

Tabela 5.8 – Esforço de Manutenção do UC-01 – Projeto A

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M* (f_{UC_E})	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_01	**15.76**	**362,13 h.h**	**53%**	**192,27 h.h**	**207,00 h.h**	**7,66%**
R&A	4.25	87,56 h.h	60%	52,53 h.h	52,00 h.h	-1,02%
Pro	2.84	58,37 h.h	50%	29,18 h.h	25,00 h.h	-16,72%
Cod	6.30	160,61 h.h	55%	88,33h.h	112,00 h.h	26,80%
Tst	2.36	55,59 h.h	40%	22,23 h.h	18,00 h.h	-23,50%

A Tabela 5.9 mostra um percentual de erro muito elevado para a etapa de projeto do UC_02. Isso ocorreu porque essa etapa precisou ser refeita. A origem dos dados necessários

para este caso de uso teve que ser alterada para corrigir algumas distorções. Por este motivo, o esforço para esta etapa foi muito acima do estimado, elevando também o percentual de erro do caso de uso UC_02. Provavelmente se um arquiteto tivesse sido alocado para realizar ou auxiliar nessa etapa, este problema não teria ocorrido.

Tabela 5.9 – Esforço de Manutenção do UC-02 – Projeto A

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M* (f_{UC_E})	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_02	**21,01**	**482,85 h.h**	**56%**	**269,03 h.h**	**320,00 h.h**	**18,95%**
R&A	5,67	116,75 h.h	65%	75,88 h.h	90,00 h.h	18,61%
Pro	3,78	77,83 h.h	45%	35,02 h.h	60,00 h.h	71,33%
Cod	8,40	214,15 h.h	60%	128,49 h.h	140,00 h.h	8,96%
Tst	3,15	74,12 h.h	40%	29,64 h.h	30,00 h.h	1,21%

Estava previsto para o Projeto A a alocação de um desenvolvedor inexperiente. Entretanto, na época da execução do UC_03, um desenvolvedor muito experiente e bastante conhecedor do sistema estava disponível e foi alocado ao projeto. Dessa forma, o esforço gasto na etapa de codificação foi bem menor que o estimado. A estimativa deveria ter sido refeita, levando-se em consideração a produtividade adequada desse desenvolvedor. Percebeu-se que não foi utilizada a recomendação do processo adotado na organização de refazer a estimativa ao longo do projeto, à medida que os requisitos fossem mais detalhados ou ocorresse alguma modificação no escopo ou na equipe alocada.

O fator de manutenção para a etapa de teste foi bem pequeno, pois o teste seria realizado pelo analista de sistemas e bastaria conferir alguns valores apresentados com um relatório já existente. Porém, esse fator de manutenção foi subestimado, o que causou um percentual de erro elevado na etapa de teste, embora pouco significativo em relação à diferença na quantidade de horas entre o previsto e o realizado. O esforço de manutenção do UC_03 é apresentado na Tabela 5.10:

Tabela 5.10 – Esforço de Manutenção do UC-03 – Projeto A

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M* (f_{UC_E})	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_03	**15,76**	**362,13 h.h**	**45%**	**161,87 h.h**	**126,00 h.h**	**-28,47%**
R&A	4,25	87,56 h.h	50%	43,78 h.h	42,00 h.h	-4,24%
Pro	2,84	58,37 h.h	60%	35,02 h.h	40,00 h.h	14,22%
Cod	6,30	160,61 h.h	50%	80,30 h.h	40,00 h.h	-100,75%
Tst	2,36	55,59 h.h	5%	2,77 h.h	4,00 h.h	44,40%

A Tabela 5.11 mostra o esforço de manutenção do UC_04. Mais uma vez, o esforço da codificação seria bem superior ao esforço estimado na etapa de teste, por essa razão a diferença no percentual do fator de manutenção:

Tabela 5.11 – Esforço de Manutenção do UC-04 – Projeto A

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M (f_{UC_E})*	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_04	**10,51**	**241,41 h.h**	**60%**	**145,39 h.h**	**160,00 h.h**	**10,05%**
R&A	2,84	58,37 h.h	50%	29,18 h.h	30,00 h.h	2,81%
Pro	1,89	38,91 h.h	50%	19,45 h.h	20,00 h.h	2,83%
Cod	4,20	107,07 h.h	80%	85,65 h.h	100,00 h.h	16,75%
Tst	1,58	37,06 h.h	30%	11,11 h.h	10,00 h.h	-11,10%

Em virtude da alteração da solução para a etapa de Projeto do UC_02, não foi mais necessária a execução da manutenção prevista no UC_05. As etapas de requisito e análise e de projeto ainda chegaram a ser iniciadas, porém elas foram descontinuadas antes de seu término e não foi mais necessário realizar as etapas de codificação e teste do UC_05. Os números mostrados na Tabela 5.12 evidenciam estas ocorrências.

Tabela 5.12 – Esforço de Manutenção do UC-05 – Projeto A

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M (f_{UC_E})*	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_05	**15,76**	**362,13 h.h**	**37%**	**133,71 h.h**	**-**	**-**
R&A	4,25	87,56 h.h	40%	35,02 h.h	20,00 h.h	-75,10%
Pro	2,84	58,37 h.h	40%	23,34 h.h	10,00 h.h	-13,40%
Cod	6,30	160,61 h.h	40%	64,24 h.h	-	-
Tst	2,36	55,59 h.h	20%	11,11 h.h	-	-

A análise de impacto revelou a necessidade de pequenos ajustes no UC_06, o que motivou a escolha de fatores de manutenção com valores baixos. E os fatores das etapas de Requsito e Análise e Projeto foram um pouco mais elevados, porque estimava-se que a codificação e o teste seriam mais simples. A Tabela 5.13 mostra que o esforço da etapa de Teste do UC_06 foi maior que o previsto. Tal fato se deu em virtude da complexidade do ambiente tecnológico envolvido, o que não foi percebido pelo analista à época da construção da estimativa. Nesse caso, o fator de manutenção dessa etapa deveria ter sido maior.

Tabela 5.13 – Esforço de Manutenção do UC-06 – Projeto A

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M* (f_{UC_E})	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_06	**10.51**	**241.41 h.h**	**24%**	**58.00 h.h**	**54.00 h.h**	**-7.41%**
R&A	2.84	58.37 h.h	30%	17.51 h.h	18.00 h.h	2.80%
Pro	1.89	38.91 h.h	30%	11.67 h.h	10.00 h.h	-16.70%
Cod	4.20	107.07 h.h	20%	21.41 h.h	16.00 h.h	-33.81%
Tst	1.58	37.06 h.h	20%	7.41 h.h	10.00 h.h	34.95%

A Tabela 5.14 apresenta o esforço total de manutenção estimado e realizado para o Projeto A. Também é mostrado o percentual de erro relativo simétrico (ERS) do Projeto A, de acordo com a *Eq.* 4.28 e a *Eq.* 4.29:

Tabela 5.14 – Esforço de Manutenção– Projeto A

Projeto A	TUCP-M	*Esforço Estimado Inicial*	*Fator-M* (f_{UC_E})	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
Etapas	**89,30**	**2.052,06 h.h**	**47%**	**960,27h.h**	**897,00 h.h**	**-7,05%**
R&A	24,11	496,17 h.h	51%	253,90 h.h	252,00 h.h	-0,75%
Pro	16,07	330,76 h.h	46%	153,68 h.h	165,00 h.h	7,37%
Cod	35,72	910,12 h.h	51%	468,42 h.h	408,00 h.h	-14,81%
Tst	13,40	315,01 h.h	27%	84,27 h.h	72,00 h.h	-17,04%

Para os seis casos de uso considerados, o esforço real total foi de 897,00 h.h, indicando um erro de -7,05 (subestimado).

Durante este e outros projetos de manutenção da organização foi verificado que não existia a especificação de alguns casos de uso ou ela estava desatualizada. Isto é, era necessário realizar uma alteração na especificação de um caso de uso já implementado, porém, sua especificação não existia ou estava desatualizada.

Caso fosse utilizada a TUCP ou a UCP seria necessário, antes de tudo, especificar o caso de uso afetado em sua totalidade, para que fossem contadas todas as transações. Entretanto, com a utilização da TUCP-M foi possível especificar somente o necessário para o entendimento da mudança, porque a TUCP-M considera apenas as transações incluídas, alteradas ou excluídas. O restante da especificação poderia ser construído em uma outra oportunidade ou iteração, tantas vezes quantas forem necessárias. Esta facilidade mostrou-se muito importante, principalmente quando se tratava de caso de uso muito extenso, que necessitaria de um tempo razoável para construir toda a sua especificação. Este tempo impactaria negativamente no cronograma do projeto e nem sempre o usuário está disposto ou pode investir muito tempo do projeto em redocumentação.

Por não possuir a especificação completa dos casos de uso e consequentemente não ser possível contar a quantidade total de transações dos casos de uso, também não foi possível calcular as estimativas do projeto utilizando a TUCP com o objetivo de comparar com a TUCP-M.

5.4 Projeto B

O Projeto B de manutenção foi realizado em um sistema de grande porte, desenvolvido na plataforma Cliente/Servidor, duas camadas, utilizando a linguagem Powerbuilder. O sistema trata de aplicações com debêntures, no mercado financeiro.

Para compor o projeto o usuário gestor selecionou um conjunto de demandas, algumas corretivas, outras evolutivas e outras de cunho legal. Seu objetivo era disponibilizar uma nova versão do sistema que atendesse às demandas legais publicadas pelos órgãos fiscalizadores, além de realizar algumas correções e melhorias.

O sistema também recebe muitas demandas durante o ano e por esta razão dispõe de um analista dedicado. Em virtude das demandas legais a serem atendidas, a organização resolveu incorporar mais um analista de sistemas na equipe. Assim, o projeto contou com 1 analista com grande experiência no sistema e um outro com pouca experiência e baixo conhecimento no sistema. O analista mais experiente ficou responsável pelos casos de uso que exigiam mais conhecimento sobre o sistema, principalmente os que apresentavam erros. O analista menos experiente ficou responsável pelos casos de uso mais simples. Também foram alocados 2 desenvolvedores: um com bastante experiência e conhecimento no sistema e um novato na organização.

A equipe foi organizada da seguinte forma: os casos de uso especificados pelo analista inexperiente seriam codificados pelo desenvolvedor experiente e os casos de uso especificados pelo analista experiente seriam implementados pelo desenvolvedor inexperiente, objetivando compensar a falta de experiência de um com a experiência do outro. Essa situação pode ser verificada facilmente na apuração do esforço dos casos de uso por etapa do ciclo de vida. No momento da execução deste projeto, não foi possível alocar um gerente de projeto. Os testes foram realizados pelos analistas.

Finalizada a análise de impacto, quando foram identificados os casos de uso, atores e transações de manutenção, ficou estabelecido que a documentação referente a etapa de projeto seria produzida posteriormente, em virtude do tempo reduzido. Assim o fator de manutenção

dessa etapa seria igual a zero. O cálculo da estimativa de tamanho do Projeto B foi realizado seguindo as etapas:

Contagem dos Atores – (UAW)

A manutenção solicitada afetou somente um ator do sistema, conforme Tabela 5.15:

Tabela 5.15 – Contagem dos Atores – Projeto B

Qtde. Atores	Complexidade	Peso	Total
1	Simples	1	1*1 = 1
			UAW = 1

Contagem dos Casos de Uso – (TUUCW)

As demandas selecionadas sensibilizaram 6 casos de uso do sistema. Para três deles foi contada 1 transação de manutenção e para um deles contaram-se 2 transações de manutenção. A Tabela 5.16 apresenta a contagem dos casos de uso:

Tabela 5.16 – Contagem dos Casos de Uso – Projeto B

Caso de Uso	Qtde. transações de manutenção	Complexidade	TUUCW
UC_01	1 t_m	Simples	5
UC_02	2 t_m	Simples	5
UC_03	1 t_m	Simples	5
UC_05	1 t_m	Simples	5
UC_06	1 t_m	Simples	5
UC_07	1 t_m	Simples	5
Total			**30**

Cálculo dos Pontos de Casos de Uso não Ajustados - (TUUCP)

Para o cálculo dos pontos de casos de uso não ajustados utilizou-se a *Eq.* 4.2:

$TUUCP = \Sigma UAW + \Sigma TUUCW$

$TUUCP = 1 + 30 \Rightarrow TUUCP = 31$

Cálculo dos Fatores de Complexidade Técnica - (TCF)

A Tabela 5.17 apresenta os valores indicados para os fatores de complexidade técnica do Projeto B. Cada nota foi indicada levando em consideração os valores mostrados na Tabela 4.4 de termos lingüísticos.

Tabela 5.17 – Fatores de Complexidade Técnica – Projeto B

Fator	Descrição	Peso	Nota	Produto
T1	Sistemas Distribuídos	2,0	0	**0**
T2	Desempenho da Aplicação	1,0	3	**3**
T3	Eficiência do Usuário-Final (*on-line*)	1,0	3	**3**
T4	Processamento Interno Complexo.	1,0	4	**4**
T5	Reusabilidade do código em outras aplicações	1,0	2	**2**
T6	Facilidade de Instalação	0,5	1	**0,5**
T7	Usabilidade (facilidade operacional)	0,5	3	**1,5**
T8	Portabilidade	2,0	0	**0**
T9	Facilidade de Manutenção	1,0	4	**4**
T10	Concorrência	1,0	3	**3**
T11	Características especiais de segurança	1,0	2	**2**
T12	Acesso direto para terceiros	1,0	0	**0**
T13	Facilidades especiais de treinamento	1,0	2	**2**
Total		-	-	**25**

Utilizando-se a *Eq.* 4.3 obteve-se o TCF do Projeto B: TCF = 0,85

Cálculo da estimativa de tamanho da manutenção - (TUCP-M)

Utilizando-se a *Eq.* 4**.4**, calcula-se o tamanho do Projeto B:

TUCP-$M_{Proj\ B}$ = 31 * 0,85 => TUCP-$M_{Proj\ B}$ = 26,35

Para calcular o tamanho da manutenção de cada caso de uso foi utilizada a *Eq.* 4.8:

$$TUCP - M_{(UC)} = \left(\frac{TUCP - M}{\sum TUUCW} \right) * TUUCW_{(UC)}$$ Onde:

TUCP-$M_{Proj\ B}$ = 26,35

$TUUCW_{UC\_01}$ = $TUUCW_{UC\_02}$ = $TUUCW_{UC\_03}$ = $TUUCW_{UC\_05}$ = $TUUCW_{UC\_06}$ = $TUUCW_{UC\_07}$ = 5

Dessa forma, o tamanho de cada um dos casos de uso do Projeto B corresponde a 4,39 TUCP-M. Como os tamanhos dos casos de uso são iguais, também seus tamanhos por etapa do ciclo de vida serão iguais. Utilizando-se a *Eq.* 4.7 esses tamanhos foram calculados, conforme apresentados na Tabela 5.18:

Tabela 5.18 – Estimativa de tamanho dos Casos de Uso por etapa do ciclo de vida–Projeto B

Caso de Uso	Requisito	Análise e Projeto	Codificação	Teste	Total
UC_01	1,19	0,79	1,76	0,66	4,39
UC_02	1,19	0,79	1,76	0,66	4,39
UC_03	1,19	0,79	1,76	0,66	4,39
UC_05	1,19	0,79	1,76	0,66	4,39
UC_06	1,19	0,79	1,76	0,66	4,39
UC_07	1,19	0,79	1,76	0,66	4,39

Conhecido o tamanho do Projeto B, pode-se estimar seu esforço, que compreende calcular os fatores de ambiente (EF), a produtividade (Prod) e por fim o esforço de manutenção relacionado ao projeto B:

Cálculo dos Fatores de Ambiente - (EF)

A Tabela 5.19 apresenta os fatores de ambiente para o Projeto B. Foram atribuídas as notas referentes a cada fator ambiental conforme Tabela 4.6 de termos lingüísticos. Nota-se que a maioria recebeu valor 3, indicando o balanceamento das experiências dos analistas e dos desenvolvedores. O cálculo do valor do fator ambiental do Projeto B utilizou a *Eq.* 4.10:

Tabela 5.19 – Fatores de Ambiente – Projeto B

Fator	Descrição	Peso	Nota	Produto
F1	Familiaridade com o processo de desenvolvimento	**1,5**	**3**	**4,5**
F2	Experiência com a aplicação em desenvolvimento	**0,5**	**3**	**1,5**
F3	Experiência com orientação a objeto	**1**	**3**	**3**
F4	Capacidade de análise	**0,5**	**3**	**1,5**
F5	Motivação	**1**	**3**	**3**
F6	Requisitos estáveis	**2**	**4**	**8**
F7	Colaboradores de meio período	**-1**	**0**	**0**
F8	Dificuldade na linguagem de programação	**-1**	**3**	**-3**
EFator				**18,5**

EFator $_{projB}$= 18,5

EF_{projB} = 1,4 + (-0,03 * 18,5) => EF_{projB} = 0,845

Cálculo da Produtividade

No projeto B não foi utilizada a *Eq.* 4.11 para calcular a produtividade de cada etapa do ciclo de vida. A produtividade foi indicada dentro do intervalo de 15 h.h a 30 h.h como explicado no Capítulo 4. A Tabela 5.20 apresenta a produtividade indicada para cada etapa:

Tabela 5.20 – Produtividade por Etapa do Ciclo de Vida – Projeto B

Etapa	Produtividade
Requisito e Análise	26
Projeto	20
Codificação	18
Teste	20

Cálculo do Esforço de Manutenção

No projeto B o esforço foi calculado utilizando-se as equações do Capítulo 4. A Tabela 5.21 mostra o esforço de manutenção do caso de uso UC_01:

Tabela 5.21 – Esforço de Manutenção do UC_01 – Projeto B

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M* (f_{UC_E})	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_01	**4,39**	**77,24 h.h**	**29%**	**22,21 h.h**	**21,00 h.h**	**-5,76%**
R&A	1,19	26,05 h.h	40%	10,42 h.h	16,00 h.h	53,55%
Pro	0,79	13,35 h.h	0%	0 h.h	0,00 h.h	0,00%
Cod	1,76	26,71 h.h	40%	10,68 h.h	4,00 h.h	-167,00%
Tst	0,66	11,13 h.h	10%	1,11 h.h	1,00 h.h	-11,00%

O caso de uso UC_01 representava a inclusão de algumas funcionalidades que exigiriam um pequeno esforço na etapa de Requisito e Análise e quase nenhum na etapa de projeto. Ficou sob a responsabilidade do analista menos experiente, que ainda sentia-se inseguro, o que ocasionou uma quantidade maior de horas para executar a etapa de Requisito e Análise. Talvez a produtividade prevista não estivesse adequada. Por outro lado, o programador mais experiente conseguiu realizar sua tarefa em um tempo bem menor que o esperado. Compensando o tempo gasto na etapa anterior. Além disso, o fator de manutenção da etapa de codificação foi superestimado, pois as alterações necessárias no código eram complexas, mas em número reduzido. Novamente os testes seriam realizados pelos analistas de sistemas o que levou a indicar um fator de manutenção baixo para essa etapa.

A Tabela 5.22 mostra o esforço de manutenção do caso de uso UC_02, o qual representa a alteração de algumas funcionalidades. Essas alterações previam um esforço considerável na etapa de requisitos, codificação e teste. Mais uma vez, a produtividade para o programador mais experiente foi subestimada.

Tabela 5.22 – Esforço de Manutenção do UC_02 – Projeto B

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M* (f_{UC_E})	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_02	**4,39**	**77,24 h.h**	**76%**	**58,68 h.h**	**51,00 h.h**	**-15,06%**
R&A	1,19	26,05 h.h	80%	20,84 h.h	22,00 h.h	5,57%
Pro	0,79	13,35 h.h	0%	0 h.h	0,00 h.h	0,00%
Cod	1,76	26,71 h.h	100%	26,71 h.h	18,00 h.h	-48,39%
Tst	0,66	11,13 h.h	100%	11,13 h.h	11,00 h.h	-1,18%

O caso de uso UC_03 refere-se a uma pequena manutenção corretiva. O esforço indicado para a etapa de requisitos foi subestimado, pois o analista julgou que diagnosticaria o erro facilmente com o entendimento dos requisitos, o que não se confirmou. A Tabela 5.23 mostra o esforço de manutenção do caso de uso UC_03:

Tabela 5.23 – Esforço de Manutenção do UC_03 – Projeto B

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M (f_{UC_E})*	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_03	**4,39**	**77,24 h.h**	**30%**	**22,94 h.h**	**24,50 h.h**	**6,80%**
R&A	1,19	26,05 h.h	30%	7,81 h.h	11,00 h.h	40,85%
Pro	0,79	13,35 h.h	0%	0 h.h	0,00 h.h	0,00%
Cod	1,76	26,71 h.h	40%	10,68 h.h	8,00 h.h	-33,50%
Tst	0,66	11,13 h.h	40%	4,45 h.h	5,50 h.h	23,60%

O caso de uso UC_05 representa uma manutenção evolutiva. Por essa razão o fator de manutenção das etapas é de 100%. Para a etapa de projeto o fator de manutenção foi nulo em virtude da decisão inicial de documentar essa etapa posteriormente. Não houve necessidade de alterações no modelo de dados. A etapa de requisito e análise foi realizada pelo analista menos experiente, o que elevou o esforço real dessa etapa. A etapa de codificação foi realizada pelo programador mais experiente, situação que exigiu um esforço real menor. Essa situação repetiu-se durante o projeto, o que leva a crer que o ideal seria calcular a produtividade separadamente para os casos de uso da responsabilidade de cada equipe. Essa dificuldade foi prevista e por essa razão tentou-se compensar a inexperiência de um com a experiência do outro. Os valores são apresentados na Tabela 5.24.

Tabela 5.24 – Esforço de Manutenção do UC_05 – Projeto B

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M (f_{UC_E})*	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_05	**4,39**	**77,24 h.h**	**83%**	**63,89 h.h**	**67,00 h.h**	**4,87%**
R&A	1,19	26,05 h.h	100%	26,05 h.h	40,00 h.h	53,55%
Pro	0,79	13,35 h.h	0%	0 h.h	0,00 h.h	0,00%
Cod	1,76	26,71 h.h	100%	26,71 h.h	16,00 h.h	-66,94%
Tst	0,66	11,13 h.h	100%	11,13 h.h	11,00 h.h	-1,18%

O analista mais experiente ficou responsável pelo caso de uso UC_06, repassando para ser implementado pelo programador menos experiente. A pouca familiaridade desse programador em ler as especificações nos padrões utilizados na organização também contribuiu para o aumento do esforço realizado nessa etapa. A Tabela 5.25 apresenta os resultados do UC_06:

Tabela 5.25 – Esforço de Manutenção do UC_06 – Projeto B

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M (f_{UC_E})*	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_06	**4,39**	**77,24 h.h**	**21%**	**16,31 h.h**	**16,50 h.h**	**1,16%**
R&A	1,19	26,05 h.h	40%	10,42 h.h	8,00 h.h	-30,25%
Pro	0,79	13,35 h.h	0%	0,00 h.h	0,00 h.h	0,00%
Cod	1,76	26,71 h.h	20%	5,34 h.h	8,00 h.h	49,81%
Tst	0,66	11,13 h.h	5%	0,55 h.h	0,50 h.h	-10,00%

O caso de uso UC_07 refere-se a implementação de novas funcionalidades e é apresentado na Tabela 5.26. Também aqui não houve necessidade de alterações no modelo de dados.

Tabela 5.26 – Esforço de Manutenção do UC_07 – Projeto B

Caso de Uso	TUCP-M	*Esforço Estimado Inicial*	*Fator-M (f_{UC_E})*	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
UC_07	**4,39**	**77,24 h.h**	**80%**	**61,66 h.h**	**54,00 h.h**	**-14,19%**
R&A	1,19	26,05 h.h	100%	26,05 h.h	20,00 h.h	-30,25%
Pro	0,79	13,35 h.h	0%	0,00 h.h	0,00 h.h	0,00%
Cod	1,76	26,71 h.h	100%	26,71 h.h	26,00 h.h	-2,73%
Tst	0,66	11,13 h.h	80%	8,90 h.h	8,00 h.h	-11,25%

Apresenta-se o esforço do Projeto B na Tabela 5.27. A experiência de um dos programadores acabou prevalecendo em relação à inexperiência do outro desenvolvedor.

Tabela 5.27 – Esforço de Manutenção do Projeto B

Projeto B	TUCP-M	*Esforço Estimado Inicial*	*Fator-M (f_{UC_E})*	*Esforço' Estimado Final*	*Esforço Real*	*ERS (erro)*
Etapas	**26,35**	**463,44 h.h**	**53%**	**245,69 h.h**	**234,00 h.h**	**-5,00%**
R&A	7,11	156,30 h.h	65%	101,59 h.h	117,00 h.h	15,17%
Pro	4,74	80,10 h.h	0%	0,00 h.h	0,00 h.h	0,00%
Cod	10,54	160,26 h.h	67%	106,83 h.h	80,00 h.h	-33,54%
Tst	3,95	66,78 h.h	56%	37,27 h.h	37,00 h.h	-0,73%

As estimativas do Projeto B foram calculadas utilizando-se a TUCP. O Resultado é apresentado na Tabela 5.28. O erro para a etapa de projeto foi muito elevado e a planilha não conseguiu mostrar o resultado.

Tabela 5.28 – Esforço de Manutenção do Projeto B – Utilizando a TUCP

Projeto B	TUCP	*Esforço Estimado Inicial*	*Esforço Real*	*ERS (erro)*
Etapas	**26,35**	**463,44 h.h**	**234,00 h.h**	**-98,05%**
R&A	7,11	156,30 h.h	117,00 h.h	-33,59%
Pro	4,74	80,10 h.h	0,00 h.h	#######
Cod	10,54	160,26 h.h	80,00 h.h	-100,33%
Tst	3,95	66,78 h.h	37,00 h.h	-80,49%

Comparando-se os valores calculados pelas duas técnicas percebe-se que em ambas as técnicas o tamanho calculado foi igual, pois a contagem dos casos de uso em ambas foi igual a 30. Isso ocorreu porque apesar da contagem das transações serem diferentes, todos os casos de uso foram classificados igualmente como de complexidade simples. Entretanto, o fator de manutenção utilizado na TUCP-M aprimorou o cálculo da estimativa, fazendo com que os valores calculados pela TUCP-M para esse projeto fossem mais próximos dos valores reais.

5.5 Projeto C

O Projeto C de manutenção foi realizado em um sistema de médio porte, desenvolvido na plataforma WEB, três camadas, utilizando a linguagem Asp.net. O sistema é responsável pela elaboração de proposta de crédito, controle e administração dos produtos Cheque e Conta Empresarial. É utilizado por meio da Intranet da organização.

O projeto compreendia a alteração de um caso de uso muito complexo, o UC_01 e a manutenção em dois outros casos de uso de menor complexidade, o UC_02 e o UC_08.

Por se tratar de sistema muito crítico para a organização, a equipe responsável por esse projeto era composta de 1 gerente de projetos, 2 analistas de sistemas, sendo um com média experiência e outro com pouca experiência. Além desses, também estavam previstos 1 arquiteto, 1 analista de teste e 3 desenvolvedores.

Realizada a análise de impacto, iniciou-se o cálculo de suas estimativas da forma seguinte:

Contagem dos Atores – (UAW)

A Tabela 5.29 apresenta a contagem dos atores do Projeto C:

Tabela 5.29 – Contagem dos Atores – Projeto C

Qtde. Atores	Complexidade	Peso	Total
6	Simples	1	1*6 = 6
2	Médio	2	2*2 = 4
2	Complexo	3	3*2 = 6
			UAW = 16

Contagem dos Casos de Uso – (TUUCW)

A Tabela 5.30 apresenta a contagem dos casos de uso do Projeto C.

Tabela 5.30 – Contagem dos Casos de Uso – Projeto C

Caso de Uso	Qtde. transações de manutenção	Complexidade	TUUCW
UC_01	13 t_m	n-complexo	20
UC_02	1 t_m	Simples	5
UC_08	1 t_m	Simples	5
Total			**30**

Cálculo dos Pontos de Casos de Uso não Ajustados - (TUUCP)

Para o cálculo dos pontos de casos de uso não ajustados utilizou-se a *Eq.* 4.2:

TUUCP = Σ UAW + Σ TUUCW => TUUCP = 16 + 30 => TUUCP = 46

Cálculo dos Fatores de Complexidade Técnica - (TCF)

A Tabela 5.31 apresenta os valores indicados para os fatores de complexidade técnica do Projeto C. Cada nota foi indicada levando em consideração os valores mostrados na Tabela 4.4 de termos lingüísticos:

Tabela 5.31 – Fatores de Complexidade Técnica – Projeto C

Fator	Descrição	Peso	Nota	Produto
T1	Sistemas Distribuídos	2,0	0	0
T2	Desempenho da Aplicação	1,0	3	3
T3	Eficiência do Usuário-Final (*on-line*)	1,0	4	4
T4	Processamento Interno Complexo.	1,0	5	5
T5	Reusabilidade do código em outras aplicações	1,0	0	0
T6	Facilidade de Instalação	0,5	0	0
T7	Usabilidade (facilidade operacional)	0,5	3	1,5
T8	Portabilidade	2,0	0	0
T9	Facilidade de Manutenção	1,0	3	3
T10	Concorrência	1,0	3	3
T11	Características especiais de segurança	1,0	0	0
T12	Acesso direto para terceiros	1,0	0	0
T13	Facilidades especiais de treinamento	1,0	2	2
Total		-	-	**21,5**

Utilizando-se a *Eq.* 4.3 obteve-se o TCF do Projeto C: TCF = 0,815

Cálculo da estimativa de tamanho da manutenção - (TUCP-M)

Utilizando-se a *Eq.* 4.4, calcula-se o tamanho do Projeto C:

$TUCP\text{-}M_{Proj\ C}$ = TUUCP * TCF

$TUCP\text{-}M_{Proj\ C}$ = 46 * 0,815 => $TUCP\text{-}M_{Proj\ C}$ = 37,49

Para calcular o tamanho da manutenção de cada caso de uso foi utilizada a *Eq.* 4.8:

$$TUCP - M_{(UC)} = \left(\frac{TUCP - M}{\sum TUUCW} \right) * TUUCW_{(UC)}$$ Onde:

TUCP-$M_{Proj\ C}$ = 37,49; $TUUCW_{UC\_01}$ = 20; $TUUCW_{UC\_02}$ = 5; $TUUCW_{UC\_08}$ = 5

Dessa forma, o tamanho do caso de uso UC_01 é de 24,99 TUCP-M e os casos de uso UC_02 e UC-08 apresentam o mesmo tamanho de 6,25 TUCP-M. Utilizando-se a *Eq.* 4.7 esses tamanhos foram calculados por etapa do ciclo de vida, conforme apresentados na Tabela 5.32:

Tabela 5.32 – Estimativa de tamanho dos Casos de Uso por etapa do ciclo de vida–Projeto C

Caso de Uso	Requisito	Análise e Projeto	Codificação	Teste	Total
UC_01	6,75	4,50	10,00	3,75	24,99
UC_02	1,69	1,12	2,50	0,94	6,25
UC_08	1,69	1,12	2,50	0,94	6,25

Conhecido o tamanho do Projeto C, é possível estimar seu esforço, que compreende calcular os fatores de ambiente (EF), a produtividade (Prod) e por fim o esforço de manutenção relacionado ao projeto C:

Cálculo dos Fatores de Ambiente - (EF)

A Tabela 5.33 apresenta os fatores de ambiente para o Projeto C. Foram atribuídas as notas referentes a cada fator ambiental conforme Tabela 4.6 de termos lingüísticos. Para calcular o valor do fator ambiental do Projeto C (EF_{projC}), utilizou-se a *Eq.* 4.10:

EFator $_{projC}$ = 12,5 => EF_{projC} = 1,4 + (-0,03 * 12,5) => EF_{projC} = 1,025

Tabela 5.33 – Fatores de Ambiente – Projeto C

Fator	Descrição	Peso	Nota	Produto
F1	Familiaridade com o processo de desenvolvimento	**1,5**	**1**	**1,5**
F2	Experiência com a aplicação em desenvolvimento	**0,5**	**0**	**0**
F3	Experiência com orientação a objeto	**1**	**4**	**4**
F4	Capacidade de análise	**0,5**	**4**	**2**
F5	Motivação	**1**	**4**	**4**
F6	Requisitos estáveis	**2**	**3**	**6**
F7	Colaboradores de meio período	**-1**	**2**	**-2**
F8	Dificuldade na linguagem de programação	**-1**	**3**	**-3**
EFator				**12,5**

Cálculo da Produtividade

No projeto C não foi utilizada a *Eq.* 4.11 para calcular a produtividade de cada etapa do ciclo de vida. A produtividade foi indicada dentro do intervalo de 15 h.h a 30 h.h como explicado no Capítulo 4. A Tabela 5.34 apresenta a produtividade indicada para cada etapa:

Tabela 5.34 – Produtividade por Etapa do Ciclo de Vida – Projeto C

Etapa	Produtividade
Requisito e Análise	22
Projeto	22
Codificação	25
Teste	22

Cálculo do Esforço de Manutenção

No projeto C o esforço foi calculado utilizando-se as equações do Capítulo 4.

O caso de uso UC_01 foi subestimado em quase todas as etapas do ciclo de vida, provavelmente em virtude de sua complexidade e pelo pouco conhecimento dos analistas em relação ao sistema. A Tabela 5.35 mostra o esforço de manutenção do caso de uso UC_01:

Tabela 5.35 – Esforço de Manutenção do UC_01 – Projeto C

Caso de Uso	TUCP-M	Esforço Estimado Inicial	*Fator-M* (f_{UC_E})	Esforço' Estimado Final	Esforço Realizado	ERS (erro)
UC_01	**24,99**	**594,32 h.h**	**52%**	**309,23 h.h**	**357,00 h.h**	**15,45%**
R&A	6,75	152,17 h.h	40%	60,86 h.h	68,00 h.h	11,73%
Pro	4,50	101,44 h.h	60%	60,86 h.h	70,00 h.h	15,02%
Cod	10,00	256,18 h.h	60%	153,70 h.h	187,00 h.h	21,67%
Tst	3,75	84,53 h.h	40%	33,81 h.h	32,00 h.h	-5,66%

Para o caso de uso UC_02 foi necessária a inclusão de um fluxo alternativo. A Tabela 5.36 mostra o esforço de manutenção do caso de uso UC_02. Apesar de sua codificação ser simples como prevista, a viabilização de seu teste exigiria um esforço bem maior. Por essa razão a grande diferença no fator de manutenção das etapas de codificação e teste:

Tabela 5.36 – Esforço de Manutenção do UC_02 – Projeto C

Caso de Uso	TUCP-M	Esforço Estimado Inicial	*Fator-M* (f_{UC_E})	Esforço' Estimado Final	Esforço Realizado	ERS (erro)
UC_02	**6,25**	**148,57 h.h**	**37%**	**55,66 h.h**	**54,00 h.h**	**-3,07%**
R&A	1,69	38,04 h.h	30%	11,41 h.h	10,00 h.h	-14,10%
Pro	1,12	25,36 h.h	70%	17,75 h.h	18,00 h.h	1,41%
Cod	2,50	64,04 h.h	15%	9,60 h.h	8,00 h.h	-20,00%
Tst	0,94	21,13 h.h	80%	16,90 h.h	18,00 h.h	6,51%

No momento da codificação do caso de uso UC_08 ocorreu mudança no escopo do projeto, o que acarretou um aumento do percentual de erro nessa etapa. O esforço de redocumentação não foi computado nesse projeto. A Tabela 5.37 mostra o esforço de manutenção do caso de uso UC_08:

Tabela 5.37 – Esforço de Manutenção do UC_08 – Projeto C

Caso de Uso	TUCP-M	Esforço Estimado Inicial	*Fator-M* (f_{UC_E})	Esforço' Estimado Final	Esforço Realizado	ERS (erro)
UC_08	**6,25**	**148,57 h.h**	**61%**	**90,82 h.h**	**98,00 h.h**	**7,91%**
R&A	1,69	38,04 h.h	60%	22,82 h.h	20,00 h.h	-14,10%
Pro	1,12	25,36 h.h	100%	25,36 h.h	25,00 h.h	-1,44%
Cod	2,50	64,04 h.h	60%	38,42 h.h	48,00 h.h	24,93%
Tst	0,94	21,13 h.h	20%	4,22 h.h	5,00 h.h	18,48%

A Tabela 5.38 apresenta o esforço de manutenção do Projeto C, o qual foi subestimado em 11,69% (subestimado). O percentual maior de erro foi exatamente na etapa de codificação, onde o escopo foi alterado.

Tabela 5.38 – Esforço de Manutenção – Projeto C

Projeto C	TUCP-M	Esforço Estimado Inicial	*Fator-M* (f_{UC_E})	Esforço' Estimado Final	Esforço Realizado	ERS (erro)
Etapas	**37,49**	**891,46 h.h**	**51%**	**455,71 h.h**	**509,00 h.h**	**11,69%**
R&A	10,12	228,25 h.h	42%	95,09 h.h	98,00 h.h	3,06%
Pro	6,75	152,16 h.h	68%	103,97 h.h	113,00 h.h	8,69%
Cod	15,00	384,26 h.h	53%	201,72 h.h	243,00 h.h	20,46%
Tst	5,62	126,79 h.h	43%	54,93 h.h	55,00 h.h	0,13%

A Tabela 5.39 apresenta o esforço de manutenção do Projeto C calculado com a utilização da TUCP. O percentual de erro foi bem mais elevado em relação à TUCP-M, pois a quantidade de transações desses casos de uso totaliza 28 transações. Número bem maior do que as 15 transações de manutenção contabilizadas, além de não levar em consideração o fator de manutenção de cada etapa do ciclo de vida. Mais uma vez, a TUCP superestimou em muito o esforço para o projeto de manutenção.

Tabela 5.39 – Esforço de Manutenção – Projeto C – Utilizando a TUCP

Projeto C	TUCP	Esforço Estimado	Esforço Realizado	ERS (erro)
Etapas	**49,72**	**1.182,17 h.h**	**509,00 h.h**	**-132,25%**
R&A	13,42	302,68 h.h	98,00 h.h	-208,86%
Pro	8,95	201,79 h.h	113,00 h.h	-78,58%
Cod	19,89	509,55 h.h	243,00 h.h	-109,69%
Tst	7,46	168,15 h.h	55,00 h.h	-205,73%

5.6 Projeto D

O Projeto D de manutenção foi realizado em um sistema de médio porte, desenvolvido na plataforma Cliente/Servidor, duas camadas, utilizando linguagem Visual Basic, que operacionaliza recebimento de propostas de crédito. A equipe prevista para esse projeto era composta de um analista de sistemas com média experiência e médio conhecimento no sistema e um programador com experiência mediana.

O cálculo da estimativa de tamanho do Projeto D foi realizado da seguinte forma:

Contagem dos Atores - (UAW)

Três atores classificados com simples foram afetados com a manutenção. A contagem dos atores é apresentada na Tabela 5.40. Para o Projeto C o UAW = 3:

Tabela 5.40 – Contagem dos Atores – Projeto D

Qtde. Atores	Complexidade	Peso	Total
3	Simples	1	3*1 = 3
			UAW = 3

Contagem dos Casos de Uso - (TUUCW)

Foram acompanhados dois casos de uso finalizados no Projeto D: UC_50, e UC_51. Obteve-se a seguinte quantidade de transações de manutenção: UC_50 = 4 t_m, UC_51 = 3 t_m. Assim, TUUCW = 15, conforme mostrado na Tabela 5.41:

Tabela 5.41 – Contagem dos Casos de Uso – Projeto D

Caso de Uso	Qtde. transações de manutenção	Complexidade	TUUCW
UC_50	4 t_m	Simples	10
UC_51	3t_m	Médio	5
Total			**15**

Cálculo dos Pontos de Casos de Uso não Ajustados - (TUUCP)

Para o cálculo dos pontos de casos de uso não ajustados utilizou-se a *Eq.* 4.2:

TUUCP = Σ UAW + Σ TUUCW TUUCP = 3 + 15 => TUUCP = 18

Cálculo dos Fatores de Complexidade Técnica - (TCF)

A Tabela 5.42 apresenta os valores indicados para os fatores de complexidade técnica do Projeto D. Cada nota foi indicada levando em consideração os valores mostrados na Tabela 4.4 de termos lingüísticos.

Tabela 5.42 – Fatores de Complexidade Técnica – Projeto D

Fator	Descrição	Peso	Nota	Produto
T1	Sistemas Distribuídos	2,0	5	10
T2	Desempenho da Aplicação	1,0	3	3
T3	Eficiência do Usuário-Final (*on-line*)	1,0	3	3
T4	Processamento Interno Complexo.	1,0	3	3
T5	Reusabilidade do código em outras aplicações	1,0	1	1
T6	Facilidade de Instalação	0,5	1	0,5
T7	Usabilidade (facilidade operacional)	0,5	3	1,5
T8	Portabilidade	2,0	0	0
T9	Facilidade de Manutenção	1,0	5	5
T10	Concorrência	1,0	2	2
T11	Características especiais de segurança	1,0	1	1
T12	Acesso direto para terceiros	1,0	0	0
T13	Facilidades especiais de treinamento	1,0	3	3
Total		-	-	33

O cálculo do TFator é realizado por: TFator = ΣFator = 33. Utilizando a *Eq.* 4.5:

TCF = 0,6 + (0,01 * TFator) => TCF = 0,93

Cálculo da estimativa de tamanho da manutenção - (TUCP-M)

Utilizando-se a *Eq.* 4.4, calcula-se o tamanho do Projeto D:

$TUCP\text{-}M_{Proj\ D} = TUUCP * TCF$

$TUCP\text{-}M_{Proj\ D} = 18 * 0,93 => TUCP\text{-}M_{Proj\ D} = 16,74$

Conhecido o tamanho do Projeto D, pode-se estimar seu esforço, da seguinte forma:

Cálculo dos Fatores de Ambiente - (EF)

A Tabela 5.43 apresenta os fatores de ambiente para o Projeto D. Foram atribuídas as notas referentes a cada fator ambiental conforme Tabela 4.6 de termos lingüísticos. Para calcular o valor do fator ambiental do Projeto D (EF_{projD}), utilizou-se a *Eq.* 4.10:

Tabela 5.43 – Fatores de Ambiente – Projeto D

Fator	Descrição	Peso	Nota	Produto
F1	Familiaridade com o processo de desenvolvimento	**1,5**	**5**	**7,5**
F2	Experiência com a aplicação em desenvolvimento	**0,5**	**5**	**2,5**
F3	Experiência com orientação a objeto	**1**	**5**	**5**
F4	Capacidade de análise	**0,5**	**5**	**2,5**
F5	Motivação	**1**	**3**	**3**
F6	Requisitos estáveis	**2**	**3**	**6**
F7	Colaboradores de meio período	**-1**	**0**	**0**
F8	Dificuldade na linguagem de programação	**-1**	**3**	**-3**
EFator				**23,5**

EFator $_{projD}$= 23,5

EF_{projD} = 1,4 + (-0,03 * 23,5) => EF_{projD} = 0,695

Cálculo da Produtividade

No Projeto D não foi utilizada a *Eq.* 4.11 para calcular a produtividade de cada etapa do ciclo de vida. A produtividade foi indicada dentro do intervalo de 15 h.h a 30 h.h como explicado no Capítulo 4. A Tabela 5.44 apresenta a produtividade indicada para cada etapa:

Tabela 5.44 – Produtividade por Etapa do Ciclo de Vida – Projeto D

Etapa	Produtividade
Requisito e Análise	25
Projeto	25
Codificação	25
Teste	20

Cálculo do Esforço de Manutenção

No projeto D o esforço foi calculado utilizando-se as equações do Capítulo 4. A Tabela 5.45 mostra o esforço de manutenção do caso de uso UC_50.

O esforço estimado para a etapa de Requisito e Análise do UC_50 foi superestimado. Deveria ter sido considerada uma produtividade de 20 para esse analista, pois ele revelava um ótimo perfil de analista de requisitos. Considerando o novo fator de manutenção, o resultado apresentaria um erro relativo de -4,70% para essa etapa.

Tabela 5.45 – Esforço de Manutenção do UC_50 – Projeto D

Caso de Uso	TUCP-M	*Esforço* Estimado Inicial	*Fator-M* (f_{UC_E})	*Esforço'* Estimado Final	*Esforço* Realizado	ERS (erro)
UC_50	**11,16**	**188,07 h.h**	**49%**	**92,86 h.h**	**102,00 h.h**	**9,84%**
R&A	3,01	52,35 h.h	50%	26,17 h.h	20,00 h.h	-30,85%
Pro	2,01	34,90 h.h	60%	20,94 h.h	36,00 h.h	71,92%
Cod	4,46	77,56 h.h	50%	38,78 h.h	40,00 h.h	3,15%
Tst	1,67	23,26 h.h	30%	6,97 h.h	6,00 h.h	-16,17%

Além disso, o esforço para a etapa de projeto foi estimado muito aquém do esforço real. Outros casos como esse ocorreram na organização, confirmando que há analistas com perfil mais adequado para requisitos e outros analistas com perfil mais adequado para projeto. O uso da TUCP-M ajudou nesse diagnóstico.

Para o caso de uso UC_51, mostrado na Tabela 5.46, o esforço maior concentrou-se na análise do projeto.

Tabela 5.46 – Esforço de Manutenção do UC_51 – Projeto D

Caso de Uso	TUCP-M	Esforço Estimado Inicial	*Fator-M* (f_{UC_E})	Esforço' Estimado Final	Esforço Realizado	ERS (erro)
UC_51	**5,58**	**94,03 h.h**	**29%**	**27,03 h.h**	**32,00 h.h**	**18,39%**
R&A	1,51	26,17 h.h	50%	13,08 h.h	21,00 h.h.	60,55%
Pro	1,00	17,45 h.h	0%	0,00 h.h	0,00 h.h.	0,00%
Cod	2,23	38,78 h.h	30%	11,63 h.h	8,00 h.h.	-45,38%
Tst	0,84	11,63 h.h	20%	2,32 h.h	3,00 h.h.	29,31%

O analista alocado tinha um ótimo perfil para analista de requisitos, mas não tinha a mesma desenvoltura na análise e projeto. Assim, deveriam ser consideradas produtividades diferentes para a mesma etapa, ou seja, uma produtividade melhor para o levantamento e especificação dos requisitos e uma produtividade pior para a análise. Essa situação repetiu-se em alguns projetos.

Ao mesmo tempo, era evidente que a quantidade de arquitetos estava insuficiente para atender todos os projetos em execução, e os analistas de sistemas acabavam absorvendo mais essa tarefa. Diante desse cenário, foi necessário realizar uma mudança nas etapas do ciclo de vida e em seu esforço correspondente. Ficou estabelecido que:

- As novas etapas do ciclo de vida e seu esforço obedeceriam ao seguinte:

  Requisito: 20%, Análise e Projeto: 25%, Codificação: 40%, Teste: 15%.
- A equipe de analistas deveria se responsabilizar pela etapa de Requisito e pela etapa de Análise e Projeto. Para isso ocorrer, a organização deveria especializar seus analistas ou no levantamento e especificação de requisitos ou na realização da análise e do projeto da aplicação, implicando em equipe de analistas dedicada à etapa de Requisitos e equipe de analistas dedicada à etapa de Análise e Projeto.
- Os arquitetos ficariam à disposição desses analistas para o suporte necessário à realização do projeto da aplicação.

A utilização da TUCP-M e a avaliação de seus resultados permitiu a detecção desse problema e a necessidade de reorganização das etapas do ciclo de vida.

Apesar dos contratempos, a estimativa geral para o projeto D foi satisfatória conforme mostrado na Tabela 5.47, ressaltando-se que o problema maior ocorreu na etapa de projeto, onde o analista era menos experiente.

Tabela 5.47 – Esforço de Manutenção do Projeto D

Projeto D	TUCP-M	*Esforço* Estimado Inicial	*Fator* (f_{UC_E})	*Esforço'* Estimado Final	*Esforço* Realizado	ERS (erro)
Etapas	**16,74**	**282,10 h.h**	**43%**	**119,89 h.h**	**134,00 h.h**	**11,77%**
R&A	4,52	78,52 h.h	50%	39,25 h.h	41,00 h.h.	4,46%
Pro	3,01	52,35 h.h	40%	20,94 h.h	36,00 h.h.	71,92%
Cod	6,70	116,34 h.h	43%	50,41 h.h	48,00 h.h.	-5,02%
Tst	2,51	34,89 h.h	27%	9,29 h.h	9,00 h.h.	-3,22%

Para os dois casos de uso considerados, o esforço real total foi de 134,00 h.h, indicando um erro de 11,77% (subestimado). O sistema relacionado ao Projeto D não possuía documentação. Por essa razão não foi possível calcular as estimativas do Projeto D utilizando a TUCP.

5.7 Percepções Durante a Utilização da TUCP-M

Durante o período de utilização da TUCP-M nos projetos de manutenção de software da organização, pôde-se observar o seguinte:

- Apesar da simplicidade de sua utilização em relação a outras técnicas existentes, é necessária a realização de treinamento sobre a TUCP-M para que os colaboradores comecem a utilizá-la.
- Além do treinamento, a experiência na utilização da técnica contribui para melhoria dos resultados apresentados.
- Os primeiros projetos que utilizaram a TUCP-M basearam-se nos seguintes percentuais de esforço por etapa do ciclo de vida: Requisito: 20%, Análise e projeto: 25%, Codificação: 45% e Teste: 10%. Entretanto, verificou-se que a maioria estava com um erro relativo simétrico acima do aceitável, nas etapas de Codificação e Teste. Diante dessa constatação, e conversando com as pessoas envolvidas nos projetos, pôde-se fazer o seguinte diagnóstico:
    - A etapa de Teste deveria ter seu percentual de esforço elevado de 10% para 15%. Em conseqüência dessa mudança, o percentual da etapa de codificação foi alterado para 40%.
    - O percentual de esforço indicado para a etapa de Codificação (40%) parecia adequado, porém os desenvolvedores não conseguiam finalizar suas tarefas no tempo estimado. Após uma investigação mais criteriosa, foi detectado que esses desenvolvedores não tinham conhecimento suficiente em UML (*Unified Modeling*

Language), que os possibilitasse ler adequadamente as especificações produzidas pelos analistas de sistemas e arquitetos. Essa dificuldade acarretava, além de atrasos nos cronogramas, problemas de relacionamento entre desenvolvedores e analistas. Substituir os desenvolvedores por aqueles que sabiam manusear e entender os documentos ou treinar aqueles que mostraram condições de atingir a produtividade esperada, mesmo que em níveis distintos, resolveu o problema.

- A utilização da TUCP-M e a divisão por etapa do ciclo de vida, favoreceram a identificação desses problemas, exigindo sua avaliação e solução para que o planejamento dos projetos possa ser construído com a confiança necessária.
- Em uma fase posterior, resolveu-se que o projeto lógico deveria ser realizado pelos analistas e o projeto físico pelos arquitetos. Assim, as etapas e seus respectivos percentuais de esforço foram redefinidos: Requisito e Análise – 27%, Projeto – 18%, Codificação – 40% e Teste – 15%. A TUCP-M foi facilmente adaptada à mudança.
- Em virtude da escassez de arquitetos, os papéis desempenhados na metodologia necessitaram ser reagrupados da seguinte forma: os analistas de sistemas ficariam responsáveis pelo levantamento de requisitos e por todo o projeto, contando com o suporte da equipe de arquitetura na etapa de projeto. Assim, os percentuais e as etapas do ciclo de vida, foram reorganizados da seguinte forma: Requisito – 20%, Análise e Projeto – 25%, Codificação – 40%, Teste – 15%. Além disso, os analistas foram especializados em analistas de requisitos e analistas de análise e projeto. Essas etapas e percentuais continuam a ser utilizados.
- A TUCP-M foi adaptada para espelhar a nova forma de trabalho, permanecendo dentro dos níveis aceitáveis de erro para estimativas dos projetos. A mudança foi realizada rápida e facilmente indicando a facilidade de uso e adaptação da técnica, caso novas mudanças dessa natureza sejam necessárias.
- Para o cálculo das estimativas de projetos de desenvolvimento, é possível utilizar a TUCP-M. Para isso, deve-se levar em consideração todas as transações do caso de uso e utilizar o fator de manutenção de todas as etapas (Fator-M ou f_{UC_Ei}) igual a 100%. Ou seja, todas as transações dos casos de uso seriam transações incluídas e todos os f_{UC_Ei}=100%.
- Percebeu-se que a TUCP-M apresenta resultados mais aprimorados para projetos de manutenção de software que a TUCP, principalmente naqueles projetos cuja quantidade de transações afetadas pela manutenção é diferente da quantidade total de transações do

caso de uso e/ou nas manutenções que não exigem que o esforço de cada etapa do ciclo de vida seja de 100%.

- A TUCP-M pôde ser utilizada em projetos de manutenção relacionados a sistemas que não tinham documentação ou a documentação encontrava-se desatualizada, o que não é possível com a TUCP. Essa situação ocorre porque a TUCP considera a quantidade total de transações dos casos de uso, por outro lado, a TUCP-M considera apenas as transações envolvidas na manutenção (incluídas, alteradas, excluídas). Essas transações são identificadas após avaliação de impacto das mudanças solicitadas e o projeto pode ser estimado sem que as demais transações dos casos de uso sejam identificadas. Ou seja, os casos de uso podem ser construídos ou atualizados parcialmente.
- Em alguns projetos que utilizaram a TUCP-M, a estimativa realizada estava muito acima ou muito abaixo dos índices de erros aceitáveis. Após estudo detalhado de como a estimativa foi realizada, constatou-se a maneira errada no uso da TUCP-M por parte dos analistas e/ou gerentes de projetos:
    - Houve erro significativo na contagem das transações de manutenção.
    - Foi levada em consideração a quantidade total de transações do caso de uso e não somente aquelas incluídas, alteradas e excluídas.
    - Os casos de uso não estavam sendo identificados corretamente.
    - Por desconhecimento, os profissionais que estavam utilizando a TUCP-M não tinham segurança suficiente para preencher os valores para o cálculo dos fatores técnicos e/ou dos fatores ambientais. Assim, optavam por repetir os valores informados para esses fatores por outros projetos. Evidentemente, essa atitude causava grande impacto na estimativa, pois na maioria dos casos, os projetos eram totalmente diferentes.
- Esses problemas levaram às seguintes conclusões e decisões:
    - Necessidade de treinamento para os analistas de sistemas e gerente de projetos na aplicação da TUCP-M e também TUCP.
    - Necessidade de realização de *mentoring* nas equipes dos projetos, para que os gerentes de projetos e os analistas de sistemas soubessem identificar os casos de uso adequadamente e contar as transações de forma correta.
    - Conscientizar sobre a importância dos fatores de complexidade técnica e dos fatores ambientais.

- Incentivar a revisão por pares tanto nas especificações dos casos de uso como na contagem das transações.

- Nas ocoasiões em que a *Eq.* 4.11, proposta para diminuir o grau de subjetividade no cálculo da produtividade, foi utilizada, produziu resultados mais aprimorados. Porém, acredita-se que ainda seja necessário realizar mais testes para atestar sua efetividade.
- Está previsto na metodologia utilizada na organização, o refinamento das estimativas durante o decorrer do projeto. Entretanto, esta prática ainda não está totalmente absorvida pelas equipes em virtude de resistências por parte do usuário gestor. Diante dessa realidade, ficou ainda mais evidente a necessidade de se ter estimativas as mais acuradas possível no início do projeto. A TUCP-M respondeu adequadamente a esta situação. Em alguns casos, quando o escopo ou a equipe mudavam bastante, a estimativa tornava-se inadequada. Entretanto, quando ela era refeita com base no novo cenário, voltava a apresentar níveis satisfatórios.
- Diante dos bons resultados apresentados na organização, foi iniciada a construção de um novo sistema, na plataforma JAVA, com base de dados em DB2 que substituirá a planilha no cálculo das estimativas dos projetos, facilitando o uso das técnicas TUCP e TUCP-M. O sistema estará em produção a partir de Julho de 2008. O novo sistema possibilitará a construção de uma base histórica da organização referente a estimativas dos seus projetos. Viabilizará também a disponibilização de diversos relatórios gerenciais e a construção de um *DataMart* sobre os valores previstos e os realizados. Esse fato abrirá um grande leque de oportunidades para que sejam avaliados os resultados registrados e consequentemente muitas decisões sobre os processos e as equipes envolvidas poderão ser tomadas.
- Além disso, essas informações permitirão cada vez mais o aprimoramento e a calibração das técnicas adotadas.
- Devido à necessidade de terceirização de alguns serviços, a organização onde o experimento foi realizado construiu e publicou 4 editais de licitação para contratar empresas para a prestação de serviços na modalidade de fábrica de software. Os editais prevêem a contratação de serviços referentes às fases de Requisitos, Análise e projeto, Codificação e/ou Teste. Todos estes serviços serão estimados e pagos utilizando-se as técnicas TUCP e TUCP-M. Os novos contratos estarão vigentes a partir de Agosto de 2008.

5.8 Conclusão

Este capítulo apresentou a proposta da técnica TUCP-M para estimar projetos de manutenção de software baseada em casos de uso. Diante dos resultados apresentados, acredita-se que a técnica venha a auxiliar fortemente o cálculo de estimativas mais acuradas para projetos dessa natureza. Verificou-se a necessidade de disseminação do conhecimento na aplicação da técnica para todos os analistas e gerentes de projetos da organização, com o objetivo de consolidar sua utilização, proporcionando a troca de experiência entre as equipes.

A contagem apenas das transações que foram incluídas, alteradas ou excluídas e a utilização do fator de manutenção por etapa do ciclo de vida do projeto, são dentre outros, fatores que auxiliam na acurácia da técnica.

Em virtude dos bons resultados apresentados, a organização continua a utilizar a TUCP-M em seus projetos de manutenção de software, como forma de avaliação e refinamento constantes da técnica e do próprio processo de desenvolvimento e evolução de software.

No Capítulo 6 serão apresentadas as considerações finais desse trabalho.

Capítulo 6
Conclusão

Este capítulo apresenta as considerações finais deste trabalho, um resumo das contribuições da TUCP-M e sugestões de trabalhos futuros.

Nas fases iniciais de um projeto de software, notadamente na fase de levantamento de requisitos, as características que definirão o tamanho do produto ainda não estão totalmente evidenciadas, o que dificulta a determinação de seu tamanho. Nesse contexto, o que é possível fazer é calcular as estimativas referentes ao projeto iniciado, com base no escopo delimitado.

As estimativas de um projeto de software representam a possibilidade das organizações realizarem o planejamento e gerenciamento desses projetos de forma mais consciente do que se pode esperar em relação aos prazos, custos e equipe envolvidos.

Várias técnicas de estimativas para projetos de software foram criadas ao longo dos anos, objetivando indicar com a menor margem de erro possível o tamanho, esforço, prazo e custo do projeto. Além dessa questão, é fundamental que a técnica utilizada ofereça condições de comparação entre o que foi previsto e o que está sendo realizado. Essa possibilidade é o que determina o controle do projeto, o qual é uma das principais atividades da gerência de projetos.

Embora existam várias técnicas que atendam a essas necessidades, a maioria é mais direcionada, ou melhor aplicada a projetos de desenvolvimento de software ou apenas às manutenções evolutivas, evidenciando uma carência dessas técnicas para projetos de manutenção de software, os quais são os mais encontrados nas organizações, principalmente para evoluir ou corrigir seus sistemas mais críticos.

Visualizando esse contexto, esse trabalho propôs a **TUCP-M** (Pontos de Caso de Uso Técnico para Manutenção de Software) que calcula estimativas para projetos de manutenção de software, a partir da abordagem de casos de uso, estendendo a técnica TUCP (Ponto de Casos de Uso Técnico), proposta para estimativas de projetos de desenvolvimento de software.

Como principais contribuições da TUCP-M apresentada nesse trabalho, pode-se destacar:

- Possibilidade de se realizar o cálculo das estimativas de tamanho, esforço, prazo e custo logo após a fase de iniciação do projeto, de forma simples e para qualquer categoria de manutenção.
- Refinamento do cálculo do tamanho do projeto de manutenção de software por considerar apenas as transações incluídas, alteradas e excluídas em virtude da solicitação de mudança.
- Aprimoramento do cálculo das estimativas para projetos de manutenção, uma vez que utiliza o fator de manutenção para cada etapa do ciclo de vida de cada caso de uso afetado, pois dependendo do tipo de manutenção (corretiva, preventiva, perfectiva, adaptativa), etapas do ciclo de vida podem ser parcialmente executadas.
- Possibilidade de se estimar projetos de manutenção que afetam sistemas sem documentação ou com sua documentação desatualizada, pois considera somente as transações incluídas, alteradas e/ou excluídas em virtude da manutenção solicitada. Essa situação permite que a estimativa de tamanho possa ser feita com os casos de uso parcialmente especificados.
- Tabela de termos lingüísticos para os fatores de complexidade técnica.
- Tabela de termos lingüísticos para os fatores ambientais.
- Utilização de documento de Solicitação de Mudança para facilitar a avaliação de impacto do projeto de manutenção.
- Indicação e avaliação dos fatores ambientais por papel mais relevante em cada etapa do ciclo de vida, objetivando diminuir a subjetividade na indicação da produtividade de cada etapa.
- Possibilidade de também estimar projetos de desenvolvimento de software.
- Viabilização da contratação de fábrica de software por etapa do ciclo de vida em projetos de manutenção.

Os resultados desse trabalho e as avaliações realizadas no decorrer da aplicação da técnica sugerem os trabalhos futuros relacionados a seguir:

- Realizar testes mais detalhados com relação à *Eq.* 4.11 que prevê a diminuição da subjetividade no cálculo da produtividade de cada etapa do ciclo de vida.
- Substituir a planilha eletrônica utilizada para os cálculos das estimativas por um sistema com base de dados centralizada, integrando-o a ferramenta de gerenciamento de cronograma, visando facilitar o acompanhamento da realização do projeto.

- Construir *Data Mart* com as estimativas relacionadas aos projetos de desenvolvimento e manutenção, com o objetivo de oferecer informações que facilitem a melhoria contínua da técnica, do processo de desenvolvimento e manutenção utilizado na organização, da equipe envolvida e o diagnóstico de possíveis problemas relacionados.
- Definir padrão de especificação de casos de uso para sistemas na plataforma mainframe e linguagem cobol, permitindo a utilização da TUCP-M no cálculo das estimativas de projetos relacionados a esses sistemas.
- Construção de ferramenta que permita calcular o tamanho da equipe necessária para atender a determinado projeto em um determinado prazo estipulado previamente.

A TUCP-M buscou o aprimoramento das estimativas para projetos de manutenção de software e como conseqüência oferecer condições mais favoráveis para o planejamento, gerenciamento e terceirização de projetos dessa natureza.

Referências Bibliográficas

(ANDA *et al*., 2001) ANDA, B. *et al*. Estimating software development effort based on use cases: experiences from industry. International Conference on the Unified Modeling Language (UML2001). Toronto, 2001.

(ANDA, 2002) ANDA, B. Comparing effort estimates based on Use Case Points with expert Estimates. Empirical Assessment in Software Engineering (EASE, 2002), Keele, UK, April, 2002.

(ABRAN, 2000) ABRAN, A. *et al*. Functional size measurement methods: COSMIC-FFP: design and field trials. FESMA-AEMES Software Measurements Conference, 2000.

(AGARWAL *et al.,* 2001) AGARWAL, R., Manish Kumar, Yogesh, S. Mallick, R.M. Bharadwaj, D.Anantwar. Estimating Software Projects. ACM, 2001.

(ANQUETIL *et al.,* 2004) ANQUETIL, N., OLIVEIRA, K. M., RAMOS, C. Conhecendo Sistemas Legados através de Métricas de Software. SBQS – Simpósio Brasileiro de Qualidade de Software, 2004.

(APRIL, 1995) APRIL, A., ABRAN, A. Industrial Research in Software Maintenance:Development of Productiviy Models. Guide Summer'95 Conference and Solutions Fair, Boston, 1995.

(BARCELLOS, 2003) BARCELLOS, M. P. Planejamento de Custos em Ambientes de Desenvolvimento de Software Orientados à Organização. (Mestrado em Engenharia de Sistemas e Computação), Universidade Federal do Rio de Janeiro – UFRJ, 2003.

(BHATT *et al*., 2004) BHATT, P., SHROFF, G., MISRA, A. K. Dynamics of Software Maintenance. ACM SIGSOFT – Software Engineering Notes, volume 29, number 5, Setembro, 2004.

(BOEHM, 1981) BOEHM, B. Software Engeneering Economics. Englewood Cliffs, NJ:Prentice Hall, 1981.

(BOEHM *et al.*, 1995) BOEHM, B., W., Clark, B., Horowitz, E. *et al*. Cost models for future life cycle processes: COCOMO 2.0. Annals Software Engineering, 1995.

(BOOCH, 1994) BOOCH, G. Object-Oriented Analysis and Design with Applications. 2nd Edition, Benjamin-Cummings Publishing Company Inc, 1994.

(BOURQUE *et al.*, 1996) BOURQUE P., MAYA M., ABRAN A. A Sizing Measure for Adaptative Maintenance Work Products. IFPUG Spring Conference, Atlanta, Abril, 1996.

(CALAZANS*,* 2005) CALAZANS, A. T. S., OLIVEIRA, M. A. L. Avaliação de Estimativa de Tamanho para projetos de Manutenção de Software. Proc. of Argentine Symposium on Software Engineering, 2005.

(CAPPELLI *et al.,* 2003) CAPPELLI, C., BORGES, M. R. S., ARAUJO, R. Qualidade do Processo de Manutenção de Software. Congresso Internacional de Tecnologia de Software, Anais CITIS 2003, Curitiba, 2003.

(CAPRETZ & CAPRETZ, 1996) CAPRETZ, L. F. & CAPRETZ, M. A. M. Object-Oriented Software: Design and Maintenance. University of Pittsburgh, USA, Series Editor S K Chang: Series on Software Engineering and Knowledge Engineering, Vol. 6, 1996.

(DAMODARAN e WASHINGTON, 2003) DAMODARAN, M., WASHINGTON, A. Estimation Using Use Case Points. Computer Science Program, Texas – Victoria; University of Houston. Disponível em: <http://bfpug.com.br/Artigos/UCP/Damodaran-Estimation_Using_Use_Case_Points.pdf>. Acesso em: 08/09/2007.

(DEMARCO, 1989) DEMARCO, T. Controle de Projetos de Software. Editora Campus, 1989.

(DUMKE, 1999) DUMKE, R.; FOLTIN, E. Applicability of full function point for Siemens AT. Proceedings of the IWSM'99, Lc Superieur, Canadá, 1999.

(FENTON, 1999) FENTON, N.; NEIL, M. Software Metrics and Risk. FESMA, Amsterdam, 1999.

(FENTON e PFLEEGER, 1997) FENTON, N., PFLEEGER, S. Software Metrics: a Rigorous & Practical Approach. PWS Publishing Company, Boston, 1997.

(FREIRE & BELCHIOR, 2006) FREIRE, Y. M. A., BELCHIOR, A. D. Estimativas de Manutenção de Software a partir de Casos de Uso. Simpósio Brasileiro de Qualidade de Software – III Workshop de Manutenção de Software Moderna, Vila Velha-ES, 2006.

(FREIRE & BELCHIOR, 2007a) FREIRE, Y. M. A., BELCHIOR, A. D. Uma Experiência em Estimativas de Projetos de Manutenção de Software. Simpósio Brasileiro de Qualidade de Software – IV Workshop de Manutenção de Software Moderna, Porto de Galinhas-PE, 2007.

(FREIRE & BELCHIOR, 2007b) FREIRE, Y. M. A., BELCHIOR, A. D. Estimates of Software Maintenance from Use Cases. SMEF 2007 – Software Measurement European Fórum - 4th edition, Roma-Itália, 2007.

(FREIRE & BELCHIOR, 2007c) FREIRE, Y. M. A., BELCHIOR, A. D. Pontos de Caso de Uso Técnico para Manutenção de Software. SBES 2007 - XXI Simpósio Brasileiro de Engenharia de Software, João Pessoa-PB, 2007.

(FPCPM, 1999) FPCPM. (1999) Function Point Counting Practices Manual, Version 4.1.

(GARMUS e HERRON, 2000) GARMUS,D., HERRON, D. Function Point Analysis – Measurement Practices for Sucessful Software Projects. Addison-Wesley Information Technology Series, 2000.

(GRUBB & TAKANG, 2003) GRUBB, P., TAKANG, A. Software Maintenance Concepts and Practice. Second edition, Singapore: Word Scientific Printers, 2003.

(HILBURN *et al*., 1999) HILBURN, T. B., HIRMANPOUR, I., KHAJENOORI, S., TURNER, R., QASEM, A. A Software Engineering Body of Knowledge. Technical Report Version 1.0, Carnegie Mellon University, Software Engineering Institute, April, 1999.

(IEEE, 1998) IEEE, Institute. IEEE Std 1219-1998. IEEE Standard for Software Maintenance. New York: Institute of Electrical and Eletronic Engineers. Inc., 1998.

(IFPUG, 2000) Count Practices Manual – Release 4.1.1, 2000, IFPUG – International Function Point Users Group.

(IFPUG, 2002) IT Measurement, A practical Advice from the Experts, 2002, IFPUG - International Function Point Users Group, Addison Wesley Information Technology Series.

(ISO/IEC 12207, 1998) ISO/IEC 12207. Tecnologia da Informação: processos de ciclo de vida de software. ABNT, Rio de Janeiro, 1998.

(ISO/IEC 14764, 1999) ISO/IEC 14764. Software Maintenance. Genebra: International Organization for Standardization, 1999.

(JALOTE, 1999) JALOTE, P. CMM in Practice: processes for executing software projects at Infosys. Addison-Wesley, 2001.

(JONES, 1994) JONES, C. Software Challenges: Function point: a new way of looking at tools. Computer, August, 1994.

(JORGENSEN; INDAHL; SJOBERG, 2003) JORGENSEN, M., INDAHL, U., SJOBERG, D. Effort Estimation: software effort estimation by analogy and regression toward the mean. Disponível em http://simula.no/research/engineering/publications/SE.4.Joergensen.2003.a. Acessado em 17/11/2007.

(KAJKO-MATTSSON *et al.,* 2001) KAJKO-MATTSSON, M., FORSSANDER, S., OLSSON, U. Corrective Maintenance Maturity Model (CMM3): Maintainer's Education and Training. Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), Toronto, Canada, 2001.

(KARNER, 1993) KARNER, G. Metrics for Objectory. Diploma thesis, University of Linköping, Sweden, December, 1993.

(KOSKINEN, 2003) KOSKINEN, J. Software Maintenance Cost Estimation and Modernization Support. ELTIS-project, University of Jyväskylä, 2003.

(KOSLOSKI, 2005) KOSLOSKI, R. A. D. Melhoria Contínua de Estimativas de Esforço para o Desenvolvimento de Software: Uma Abordagem sobre Produtividade. Universidade Católica de Brasília, Dissertação de Mestrado, 2005.

(KUSTERS, 2001) KUSTERS, R. J., HEEMSTRA, F. J. Software Maintenance: an approach towards control. IEEE International Conference on Software Maintenance (ICSM'01), 2001.

(LEHMAN, 1996) LEHMAN, M. M. Feedback in the software evolution process. Keynote Address, CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics, Dublin, Information and Software Technology 38, Special Issue on Software Maintenance, 1996.

(LEHMAN *et al.,* 1998) LEHMAN, M. M, RAMIL, J. F., WERNICK, P. D., PERRY, D. E. Metrics and Laws of Software Evolution – The Nineties View. IEEE International Conference on Software Maintenance, 1998.

(LONGSTREET, 2002) LONGSTREET, D. Fundamentals of Function Point Analysis. Blue Springs: Longstreet Consulting Inc., 2002.

(MCGARRY *et al.*, 2002) MCGARRY, J., CARD, D., JONES, C., LAYMAN, B., CLARK, E., DEAN, J., HALL, F. Practical Software Measurement (PSM) – Objective Information for Decision Makers. Addison Wesley, 2002.

(MCPHEE, 1999) MCPHEE, C. S. Software process management: software size estimation. University of Calgary, 1999.

(MENESES, 2001) MENESES, J. B.; MOURA H. P. Inspector: Um Processo de avaliação de progresso para projetos de software. Dissertação de Mestrado, Universidade Federal de Pernambuco (UFPE), Recife, 2001.

(MISIC e TESIC, 1998) MISIC, V., TESIC D. Downsizing the estimation of software quality: a small objectoriented case study. Technology of Object-Oriented Languages and Systems, 1998.

(MÖLLER e PAULISH, 1993) MOLLER, K. H., PAULISH, D. J. Software Metrics, A Practitioner's Guide to Improved Product Development. IEEE Computer Society Press, Chapman & Hall, 1993.

(MONTEIRO, 2005) MONTEIRO, T. C. Pontos de Caso de Uso Técnicos (TUCP): Uma Extensão da UCP. Universidade de Fortaleza, Dissertação de Mestrado, 2005.

(MYLIUS; MARODIN, sd) MYLIUS, S.; MARODIN H. O estabelecimento de estimativas em projetos. S.d.

(NESMA, 1996) NESMA FPA Counting Practices Manual (CPM 2.0). Netherlands Software Metrics Users Association, NESMA, 1996.

(NESMA, 2005) The Estimated Function Point Count. Disponível em <http://www.nesma.nl/english/index.htm>. Acessado em novembro de 2007.

(PARTHASARATHY, 2007) PARTHASARATHY, M. A. Practical Software Estimation – Function Point Methods for Insourced and Outsourced Projects. Paperback, 2007.

(PETERS, 1999) PETERS, K. Software project estimation. software productivity. Vancouver, British Columbia, Canadá.1999. Disponível em: <http://www.spd.ca/downloads/resources/estimates/estbasics.pdf>. Acesso em 12/11/2004.

(PFLEEGER, 2007) PFLEEGER, S. L. Engenharia de Software : Teoria e Prática. Tradução Dino Franklin; revisão técnica Ana Regina Cavalcanti da Rocha; 2ª. Edição, SãoPaulo, Prentice Hall, 2007.

(PIGOSKY, 1996) PIGOSKY, T. Pratical Software Maintenance – Best Practices for Managing Your Software Investiments. Wiley Computer Publisinhg, Nova York, 1996.

(PMBOK, 2004) PMBOK - PROJECT MANAGEMENT BODY OF KNOWLEGDE, Belo Horizonte: PMI- MG, (edição traduzida pelo PMI/MG). 2004.

(POLLICE, 2004) POLLICE, G. Measuring Up. Copyright IBM Corporation 2004 – RationalEdge, 2004.

(POTOK e VOUK, 1999) POTOK, T. E., VOUK M. A. A Model of Correlated Team Behavior in a Software Development Environment. IEEE, 1999.

(PRESSMAN, 2000) PRESSMAN, R. Software Engineering: A Practioner´s Approach. McGraw-Hill, 5ª edição, 2000.

(PUTNAM e MAYERS, 2003) PUTNAM, L. H., MAYERS, W. Five Core Metrics – The Intelligence Behind Sucessful Software Management. Dorset House Publishing, 2003.

(RAMIL e LEHMAN, 2000) RAMIL, J. F., LEHMAN, M. M. Cost Estimation and Evolvability Monitoring for Software Evolution Process. Workshop on Empirical Studies of Software Maintenance, San Jose, CA, USA, 2000.

(RAMOS *et al*., 2004) RAMOS, C. S., OLIVEIRA, K. M., ANQUETIL, N. Legacy Software Evaluation Model for Outsourced Maintainer. Proceedings of the 8th European Conference on Software Maintenance and Reengineering (CSMR 2004), Tampere, Finland, IEEE Computer Society, May, 2004.

(RIBU, 2001) RIBU, K. Estimating Object-Oriented Software Projects with Use Cases. Master of Sciense Thesis, University of Oslo, Norway. November, 2001.

(ROSS, s.d) ROSS, M. Size does matter: continuous size estimating and tracking. QSM - Quantitative Software Management. Disponível em http://www.qsm.com. Acessado em Novembro/2007.

(RUP, 2003) Rational Software Corporation, Rational Unified Process. Version 2003.06.00.65, CD-ROM, Rational Software, Cupertino, California, 2003.

(SIMÕES, 1999) SIMÕES, C. Sistemática de Métricas, Qualidade e Produtividade. Developers' Magazine, Brasil, 1999.

(SIMON, 2000) SIMON, I. K. Using function point analysis method or line of code for software size estimation?. Proceedings of ESCOM-SCOPE 2000 Conference, Munique, Germany, 2000.

(SNEED, 1995) SNEED, H. M. Estimating the Costs of Software Maintenance Tasks. 11th International Conference on Software Maintenance (ICSM'95), 1995.

(SOMMERVILLE, 2007) SOMMERVILLE, I. Software Engineering. 8a. Edição, Addison-Wesley, 2007.

(SWEBOK, 2004) SWEBOK, 2004 Guide to the Software Engineering Body of Knowledge. IEEE, 2004 version.

(TRAN-CAO e LEVESQUE, 2002) Tran-Cao, D., Levesque, G. Maintenance effort and cost estimation using software functional sizes. University of Quebec at Montreal (UQAM), 2002.

(VAZQUEZ *et al*., 2007) VAZQUEZ, C., SIMÕES, G., ALBERT, R. Análise de Pontos de Função, Medição, Estimativas e Gerenciamento de Projetos de Software. 7a. ed, São Paulo, Erica, 2007.

(WEBSTER *et al.,* 2004) WEBSTER, K. P. B., OLIVEIRA, K., M., Anquetil N. Priorização de Riscos para Manutenção de Software. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, Madrid, Espanha, 2004.

Modelo de Solicitação de Mudança

Sistema SXXX – <Nome do sistema>

Solicitação de Mudança

CONTEÚDO

1. Histórico de Revisões

Data	Versão	Descrição	Autor
<dd/mm/aaaa>	<x.x>	<detalhes>	<nome>

2. Identificação

2.1 Finalidade

Documentar e acompanhar a solicitação de mudanças em projetos.

2.2 Referências

[Referencie todos os documentos mencionados ao longo deste documento.]

3. Identificação da Solicitação de Mudança

3.1 Título

[Informe um título que identifique a mudança.]

3.2 Solicitante

[Nome do solicitante.] - *[Unidade organizacional.]* - *[Fone ou e-mail.]*

3.3 Classificação da Solicitação

Prioridade (Alta, Média, Baixa)	**Data de Submissão**	**ID da Solicitação de Mudança**	**Tipo** (Correção, Melhoria)
[Prioridade.]	*[Data submissão.]*	*[Código da solicitação no sistema.]*	*[Tipo de mudança.]*

3.4 Data limite da Implantação da Mudança (em caso de demanda Legal)

[Data Limite.]

4. Detalhamento da Mudança

Problemas que originaram a mudança
[Informe os problemas que originaram a mudança, se for o caso.]

Descrição da mudança
[Descreva a mudança necessária.]

Funcionalidade a incluir	Funcionalidade a alterar	Funcionalidade a excluir
[Liste as funcionalidades a incluir.]	*[Liste as funcionalidades a alterar.]*	*[Liste as funcionalidades a excluir.]*

Impactos da mudança
[Apresente os impactos da solicitação da mudança.]

Sugestões para Contagem das Transações

O que deve ser contado com uma transação:

- Passos do fluxo de eventos do caso de uso que contenham campos de entrada possuindo valores passíveis de escolha originados de leitura de dados (listas de opções, *combos* e *grids*), por exemplo: “A aplicação exibe os nomes dos assinantes de uma companhia telefônica”;
- Passos do fluxo de eventos do caso de uso que apresentem retorno de consultas com filtros preenchidos por buscas em bancos de dados, por exemplo: “A aplicação exibe uma lista contendo os nomes da rua de acordo com o CEP selecionado”;
- Passos do fluxo de eventos do caso de uso que proporcionem validações complexas de negócio (relacionamentos com outras entidades, verificações de existência, etc.);
- Passos do fluxo de eventos do caso de uso que contenham uma geração de relatório são considerados como uma transação, e cada filtro das consultas apresentados no relatório será considerado uma outra transação; assim, a quantidade de filtros será a quantidade de transações;
- Passos onde é necessária a transformação de extensões entre arquivos. Por exemplo: passar um arquivo HTML passar para XML;
- Passos do fluxo de eventos do caso de uso que apresentem funcionalidades de consultas auxiliares como casos de uso a parte (extensão), por exemplo, telas *pop-up*;
- Fluxos alternativos do caso de uso que tenham uma quantidade de validações maior ou igual a 5, contam-se como transação e se a quantidade de validações for maior que 10, contam-se uma transação a cada grupo de 5 validações;
- Passos do fluxo de eventos do caso de uso onde existirem validações simples (obrigatoriedade, formato, etc.) de campo de entrada de dados são considerados como uma única transação se a quantidade de validações for menor ou igual a 10; se a quantidade de validações for maior que 10, conta-se uma transação a cada grupo de 5 validações. Isto foi obtido através de experimentos realizados nos ativos da organização onde se realizou o estudo de caso.

O que não deve ser contado como transação:

- Passos do fluxo de eventos que descrevam o início e o fim do caso de uso, por exemplo: “O caso de uso se inicia quando o usuário abre uma tela de filtro” ou “O caso de uso se encerra”;

- Passos do fluxo de eventos que detalhem a interação entre o sistema e o ator, por exemplo, “O usuário pressiona confirmar” ou “o sistema solicita ao usuário informar a operação (incluir, alterar, excluir)”;
- Passos do fluxo de eventos que solicitem escolhas com valores fixos (sem leitura de dados);
- Passos do fluxo de eventos que façam leituras auxiliares de dados que já tenham sido realizadas em outros fluxos do mesmo caso de uso; e fluxos alternativos do caso de uso que são apenas validações simples e que contenham mensagens de erro. Por exemplo: “Se após o passo 5 do fluxo básico, ocorrer um erro de acesso à base de dados, a aplicação exibe a mensagem ‘Erro no acesso à base de dados’”.

Sistema S123 – Risco de Crédito

Especificação de Caso de Uso

UC01 - Calcular LRC Automático

CONTEÚDO

HISTÓRICO DE REVISÕES

Data	Versão	Descrição	Autor
26/07/2005	1.0	Criação do documento	Analista 1
10/06/2007	1.1	Manutenção	Analista 1

Especificação de Caso de Uso

1. Características Gerais do Caso de Uso

1.1 Nome

Calcular Limite de Risco Cliente

1.2 Descrição

Este caso de uso é responsável por recuperar os dados de cada cliente do sistema de cadastro de clientes do BNB e baseado nas regras definidas no Documento de Regras de Negócio efetuar o cálculo do limite de risco para estes clientes por tipo de limite. Os tipos de limite utilizados serão os mesmos cadastrados no sistema.

1.3 Atores

Tempo, SXXX

1.4 Precondições

Os tipos de limite devem estar cadastrados. Os parâmetros utilizados no cálculo (percentual de utilização, percentual de comprometimento da renda bruta) devem estar cadastrados.

1.5 Pós-condições

Os clientes selecionados por todos os filtros definidos nas regras de negócio possuirão um limite de risco disponível para ser aprovado pelo gerente.

2. Fluxos de Eventos

2.1 Fluxo Básico

1. O caso de uso se inicia com a execução do componente de cálculo em uma determinada hora previamente programada. **RN2.1.1; RN2.1.2**
2. O sistema inicia o cálculo do LRC para os clientes Pessoa Física executando o sub-fluxo: Cálculo para Pessoa Física;
3. O caso de uso se encerra.

Obs.: não houve nenhuma transação de manutenção **(TM)** excluída.

2.2 Sub-fluxos

2.2.1 Cálculo para Pessoa Física

1. Este sub-fluxo se inicia quando o componente executa o cálculo para clientes do tipo Pessoa Física;
2. O sistema recupera as informações cadastrais dos clientes aplicando os devidos filtros. [E1] [E2] **RN2.1.3** **(1T= 1 transação)**

3. O sistema recupera as variáveis a serem utilizadas no cálculo; [E1] **RN2.1.4 (1T) (1TM)**
4. O sistema calcula as variáveis a serem utilizadas no cálculo; **RN2.1.5 (1T) (1TM)**
5. O sistema calcula os limites dos clientes; **RN2.1.6 (1T) (1TM)**
6. O caso de uso retorna para o passo 3 do fluxo básico.

2.3 Fluxos Alternativos

2.4 Fluxos de Exceção

2.4.1 E1 – Falha desconhecida

1. Este fluxo se inicia quando uma exceção não tratada é capturada pelo sistema.
2. O caso de uso propaga a mensagem de erro.
3. O caso de uso se encerra.

2.4.2 E2 – Registro de Inconsistências

1. Este fluxo se inicia quando o cliente a ser calculado possui inconsistências em seu cadastro. **RN2.2**
2. O sistema registra as inconsistências / ocorrências do cliente. **(1T)**
3. O caso de uso retorna ao passo 2 do fluxo básico.

3. Pontos de Extensão

Não se aplica.

4. Relacionamento com Outros Casos de Usos

4.1 Casos de Uso de Inclusão

Não se aplica.

4.2 Casos de Uso de Extensão

Não se aplica.

4.3 Generalização

Não se aplica.

5. Requisitos Especiais

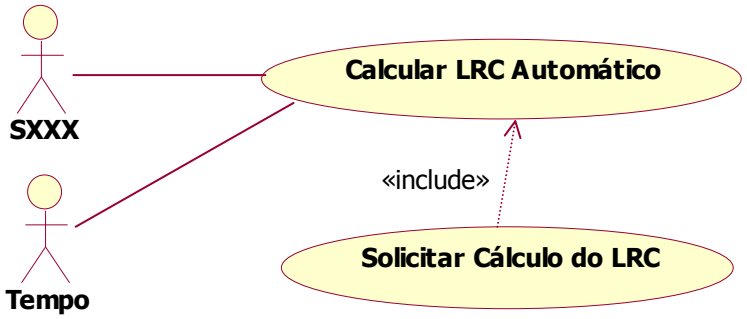
5.1 Calcular Automaticamente

O cálculo do LRC deve ser realizado automaticamente e periodicamente sem intervenção do usuário.

5.2 Valores utilizados pelo cliente

O sistema deve considerar que o valor utilizado do cliente foi recuperado com sucesso se e somente se todos os sistemas provedores das informações de débito do cliente retornarem as informações com sucesso. Caso um dos sistemas falhe ao retornar as informações o cálculo não deve ser realizado para este cliente.

6. Diagrama de Caso de Uso



7. Outras Questões

ID	Descrição	Data limite	Responsável

Apêndice D

Solicitação de Mudança para o Sistema Risco de Crédito

Sistema S123 – Risco de Crédito

Solicitação de Mudança

CONTEÚDO

1. Histórico de Revisões

Data	Versão	Descrição	Autor
04/06/2007	1.1	Formalizar manutenção 01	Maria

2. Identificação

2.1 Finalidade

Documentar e acompanhar a solicitação de mudanças em projetos.

2.2 Referências

Normas definidas pela Ambiente de Normas de Crédito.

3. Identificação da Solicitação de Mudança

3.1 Título

Alteração no cálculo do LRC e criação de portfólio de limites por produto.

3.2 Solicitante

João Carlos - Ambiente de Normas de crédito - 3129

3.3 Classificação da Solicitação

Prioridade (Alta, Média, Baixa)	**Data de Submissão**	**ID da Solicitação de Mudança**	**Tipo** (Correção, Melhoria)
Média	*05/06/2007*	*SOL-0123*	*Melhoria*

3.4 Data limite da Implantação da Mudança (em caso de demanda Legal)

Não se aplica

4. Detalhamento da Mudança

Problemas que originaram a mudança
O Cálculo do LRC não está aprimorado o suficiente para se avaliar o risco correspondente. Além disso, é necessário que os limites sejam disponibilizados por produto.

Descrição da mudança		
Nova forma de cálculo do LRC. *Disponibilizar portfólio por produto.*		
Funcionalidade a incluir	**Funcionalidade a alterar**	**Funcionalidade a excluir**
1.Calcular e disponibilizar sob demanda o portfólio de limites de risco disponíveis para concessão de crédito e posterior contratação de produtos de crédito.	*1. No cálculo da variável Percentual de Utilização do Limite, utilizar novos percentuais por produto disponível no manual de normas.* *2. Alterar o cálculo da variável Fator de Risco Cliente conforme fórmula abaixo:* $FRC = \left(\frac{Média\ (NRC_{(ant)}, NRC_{(atual)}}{10} \right)$ *3. Incluir novas famílias de limites e produtos considerando nova fórmula para o limite total:* $LT = \sum_{j=1}^{m} LF_{jj}$	*Nihil*

Impactos da mudança
O novo cálculo do LRC será realizado de forma mais aprimorada, facilitando a avaliação de risco do cliente. *A disponibilização de portfólio por produto facilitará a avaliação dos limites de crédito.*

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo