FLÁVIO CORRÊA MAIA DE CARVALHO

Diretrizes para utilização de um modelo de estimativa de esforço para sistemas web.

Monografia apresentada à Escola Politécnica da Universidade de São Paulo, para conclusão do Curso de MBA em Engenharia de Software.

Área de Concentração:

Engenharia de Software.

Orientador:

Profª. Jussara Pimenta Matos

São Paulo

2003

FICHA CATALOGRÁFICA

Carvalho, Flávio Corrêa Maia de
Diretrizes para utilização de um modelo de estimativa de esforço para sistemas para Internet. São Paulo, 2003.
XXX p.

Monografia de conclusão de curso – Escola Politécnica da Universidade de São Paulo. MBA Engenharia de Software.

1.Engenharia de Software 2. Métricas de Software 3. Estimativa de Custo.

AGRADECIMENTOS

À minha orientadora e Profª Jussara Pimenta Matos pelas diretrizes apontadas, pela paciência e pela força até o último minuto.

A Scale Systems, pelo apoio e reconhecimento incondicionais.

À minha família, por proporcionar os caminhos sólidos para minha formação.

RESUMO

Os projetos de desenvolvimento de software, na maioria das vezes, não atendem aos prazos nem aos orçamentos, de acordo com o planejamento previamente realizado. Nesse trabalho, são apresentadas diretrizes básicas para a implantação de um programa de métricas que pode auxiliar as organizações na melhoria, tanto da qualidade quanto do processo de desenvolvimento de software. O uso de modelos de estimativas de esforço auxilia o controle dos custos de um projeto e facilita sua gestão, à medida que permite uma previsão mais apurada do esforço a ser despendido bem como, do tempo necessário para a execução das diferentes atividades que envolvem o desenvolvimento de um produto.

ABSTRACT

Most of the software projects overrun schedule and cost previously planned. This dissertation introduces basic guidelines for a measurement program adoption which can help software organizations improve their software development process and the product quality. The use of estimation models in a measurement program scope contributes to a better control of costs and improvement of the project management since it provides ways to predict more accurately the effort and development time needed.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ASP	Active Server Pages
COCOMO	Constructive Cost Model
B2B	Business-to-business
B2C	Business-to-consumer
CGI	Common Gateway Interface
CMM	Capability Maturity Model
COTS	Commercial of the Shelf
EAP	Estrutura Analítica do Projeto
EDI	Electronic Data Interchange
EM	Multiplicadores de Esforço
ERP	Enterprise Resource Planning
HH	Homem-Hora
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
J2EE	Java 2 Enterprise Edition
LOC	Lines of Code
Mb	Megabytes
NS	Nominal Schedule
PF	Ponto de Função
PHP	Personal Home Pages
PM	Esforço estimado em Homem-hora
SF	Fatores de Escala
SGML	Standard Generalized Markup Language
SHTML	Server-side HTML
SLOC	Source Lines of Code
TCP/IP	Transfer Control Protocol / Internet Protocol
VRML	Virtual Reality Modeling Language
WEB	World Wide Web
XML	Extensible Markup Language

CAPÍTULO 1

1 INTRODUÇÃO

1.1 Objetivo

Esse trabalho visa proporcionar um conjunto de diretrizes que estabelecem condições básicas, para a implantação de um programa de métricas em organizações que desenvolvem e realizam manutenção de software.

1.2 Motivação

As motivações iniciais deste trabalho decorrem das dificuldades no dimensionamento de projetos e do esforço empregado na implantação destes na empresa do autor deste trabalho. Esse tem sido fator crítico na maioria das negociações comerciais, onde o valor do projeto é pré acordado, pois o esforço realizado compreende um custo maior do que o estimado.

1.3 Justificativa

Geralmente, em projetos de desenvolvimento de software, existem dificuldades a serem enfrentadas na definição dos requisitos do produto a ser desenvolvido. Os contratantes não têm esses requisitos claramente definidos, e conseqüentemente, não têm um processo de aquisição de software para garantir que a qualidade do produto adquirido seja satisfatória. Sua forma de contratação, geralmente, consiste em solicitar um orçamento com valores fixos, não variáveis com o esforço utilizado. Com essas condições, torna-se imperativo que o esforço empregado seja bem estimado.

Contudo, é comum observar que quem contrata não explicita suas expectativas em relação a determinadas características ou funções desejadas no momento da análise

de negócios, em que os requisitos são inicialmente definidos, deixando para as fases de análise, construção e até implantação do projeto.

A utilização de protótipos auxilia a estabelecer e a elucidar as funções do produto a ser construído, e também, a dimensionar o esforço e os recursos necessários. Mas, apesar de representar, reconhecidamente, um esforço extra em prol de uma maior segurança nas negociações, a utilização de protótipos não resolve esse aspecto totalmente.

Desta forma, a necessidade de se utilizar um processo de métrica visa, não só melhorar as estimativas iniciais e pouco precisas, mas também, medir o esforço extra utilizado na construção de protótipos e no atendimento de requisitos que são explicitados nas fases posteriores à análise. A formação de um histórico pode fornecer informações desse esforço extra que é realizado e não remunerado.

Os projetos de desenvolvimento de software voltados para o mercado Web são projetos cujo consumidor tem perfis diferenciados. A utilização de modelos de estimativas voltados para esse mercado pode auxiliar no processo de desenvolvimento ainda pouco maduro.

1.4 Metodologia de Trabalho

Esse trabalho apresenta conceitos relativos à utilização de métricas no processo de desenvolvimento e de manutenção de software, para dar suporte a apresentação dos modelos de estimativas avaliados, visando apoiar as diretrizes básicas para a implantação de um programa de métricas.

As diretrizes foram obtidas através do estudo dos modelos de estimativa, comparando-se com as metodologias apresentadas em publicações importantes voltadas para organizações, cujo nível de maturidade do CMM não seja superior a 2.

O capítulo 2 apresenta conceitos referentes aos modelos de estimativa, os seus objetivos, as formas de cálculo e finaliza com uma comparação entre os modelos abordados. No capítulo 3 são apresentados tópicos sobre o histórico de projetos, isto é, sua formação e interpretação. No capítulo 4 são abordadas as considerações em relação à complexidade de sistemas e sua contribuição no processo de desenvolvimento. No capítulo 5 são apresentadas as diretrizes básicas para a implantação de um programa de métricas na organização. Uma discussão a respeito da aplicação de um modelo de estimativa, em sistemas baseados em Web, é apresentada no capítulo 6, considerando as particularidades deste tipo de sistema e, por fim, as conclusões que compõem o capítulo 7.

CAPÍTULO 2

2 MODELOS DE ESTIMATIVA

2.1 Introdução

Neste capítulo, são apresentados os principais modelos de estimativas utilizados em organizações que desenvolvem e mantém aplicações de software. Esses modelos são elaborados a partir da análise e da generalização empírica de informações observadas durante o processo de desenvolvimento de software, cuja natureza é quantitativa, dessa forma, permite a utilização de métodos para medi-las. Portanto, a aceitação e a validação de um modelo ocorrem, não somente através de sua comprovação empírica e de demonstrações matemáticas, mas também com a comprovação realizada pelas organizações que a adotam, podendo até se tornar um padrão. Os principais objetivos destes modelos são a melhoria do processo de desenvolvimento e da qualidade do software desenvolvido.

2.2 Conceitos básicos

O termo **software** aplicado nesse trabalho, de acordo com Melnikoff (2002), é definido como:

1. Um conjunto de instruções de um programa de computador que, quando executado, realizam as funções especificadas, com o desempenho esperado;
2. Um conjunto de dados que permite aos programas manipularem adequadamente a informação.
3. Um conjunto de documentos que descrevem a operação e o uso dos programas.

Neste trabalho, o **produto** significa o software desenvolvido por profissionais que seguem um processo de desenvolvimento. Um processo de desenvolvimento (MELNIKOFF, 2002) é importante para:

- Definir as atividades a serem conduzidas no projeto;
- Uniformizar o entendimento dos envolvidos em relação ao desenvolvimento de sistemas;
- Manter a consistência entre sistemas desenvolvidos em uma mesma empresa;
- Viabilizar pontos de controle para a gerência.

Neste trabalho, **projeto de software** é o conjunto de todos os artefatos resultantes e necessários para a iniciação e finalização de cada uma das fases de um processo de desenvolvimento.

Conforme define Silveira (1999) apud (IEEE Std 610.12-1990), **métrica** é uma medida quantitativa do grau que um sistema, componente ou processo possui em relação a um determinado atributo. Essas indicações são relativas às características do projeto, ao tamanho do produto, ao esforço e ao prazo requerido e à qualidade do produto. As diretrizes deste trabalho consideram a definição de **estimativa**, segundo Aguiar (2002), como "a projeção quantitativa das características dos projetos".

2.3 Abordagens utilizadas

Há várias abordagens (BOEHM, 1981) que podem ser usadas em um processo de estimativa de esforço em desenvolvimento de software. De maneira geral, elas são divididas em: as que utilizam a experiência de especialistas e as que dependem da informação quantitativa do tamanho do projeto, como o número de linhas de código. Em ambos os casos, realizam-se comparações com medidas ou padrões de referência, cujas grandezas, a serem estimadas, são conhecidas.

Boehm (1981) discute cinco modelos, baseados nestas abordagens, que deram origem aos modelos de estimativas apresentados neste capítulo:

- **Modelo Algorítmico ou Paramétrico**

O modelo algorítmico requer o uso de equações baseadas em informações e dados históricos de projetos, tais como, números de linhas de código ou números de pontos de funções.

Esse modelo considera, não apenas o tamanho do projeto, mas também fatores que podem influenciar na produtividade e conseqüentemente nas estimativas de esforço e tempo de desenvolvimento. Alguns desses fatores, considerados importantes, são os níveis de experiência dos profissionais, a metodologia utilizada no processo de desenvolvimento, a linguagem de programação utilizada e a plataforma adotada.

Esses fatores, denominados **parâmetros**, representam uma relação matemática entre o tamanho do projeto, o esforço para realizá-lo, o prazo e a qualidade desejada (AGUIAR, 2002).

A equação básica, demonstrada pela eq.(2.1), que rege o cálculo de esforço para esse modelo é:

$$E = A + B \times (ev)^C \qquad (2.1)$$

onde A, B e C são constantes empiricamente obtidas, E é o esforço e *ev* é o valor do número de linhas de código ou número de pontos de função estimados. No modelo simples de Boehm (1981), o ajuste dessas constantes conduz à eq. (2.2).

$$E = 3.2 + (KLOC)^{1.05} \qquad (2.2)$$

Enquanto a eq. (2.2) é baseada em linhas de código, a eq. (2.3) exibe o modelo orientado a pontos de função de Albrecht e Gaffney (1979) apud Pressman (2001).

$$E = -91.4 + 0.355\ PF \qquad (2.3)$$

Esse método proporciona a geração de resultados com um certo grau de facilidade. Os valores podem ser calculados rapidamente, e também, podem ser

armazenados em planilhas ou em bancos de dados. Caso o histórico de projetos não esteja disponível, ainda é possível recompor as informações de esforço efetivamente utilizado, assim como, coletar o número de linhas de código ou pontos de função realizados. Desta forma, é possível modificar as informações de entrada, para refinar e ajustar o modelo, adaptando as fórmulas utilizadas.

Os resultados podem ser questionáveis, quando não são analisados os parâmetros que tem influência sobre a produtividade. Por exemplo, a utilização de novas tecnologias e a presença de condições não semelhantes às encontradas no histórico de projetos, podem influenciar negativamente as estimativas obtidas.

Minimizar esses problemas exige que o modelo seja calibrado, conforme os parâmetros julgados importante para cada organização.

- **Modelo de Estimativa por Analogia**

Estimar por analogia significa, comparar o projeto proposto com dados históricos de projetos similares, cujas informações são conhecidas. Os dados comumente avaliados neste método são o número de linhas de código, o tempo de realização do projeto, o tamanho da equipe, as despesas adicionais, além de considerações a respeito de experiências em projetos anteriores, que auxiliam a minimizar riscos.

Assim, as estimativas são baseadas nos dados do projeto atual e em outros já finalizados, de tal forma que, é possível facilmente identificar as diferenças e avaliar o impacto no novo projeto para que este possa ser estimado.

Identificar as diferenças e semelhanças entre projetos de forma a determinar o nível de similaridade é, porém, uma tarefa delicada. Há que se investigar a adequação das informações históricas para uma comparação com o projeto atual, visto que um projeto nunca seria em tudo similar a outro.

- **Modelo de Julgamento de Especialistas**

Este modelo envolve a contribuição de especialistas para estimar, através de suas experiências em diferentes tipos de projetos, o tempo e o custo de um empreendimento, além de facilitar o entendimento do projeto para o restante da equipe, de tal forma a convergirem em direção a estimativas mais consistentes.

Cada especialista, através de sua própria experiência, pode deliberar sobre os requisitos do projeto em questão, inferindo o impacto causado pelo novo projeto, a tecnologia e a metodologia adotada, sem eliminar, no entanto, o grau de incerteza próprio do trato de projetos semelhantes, mas nem sempre idênticos. Há que se contar com a inclusão de elementos específicos ou inovadores, avaliando as circunstâncias caso a caso. Não existem, portanto, regras absolutas e imutáveis, sendo um desafio permanente no planejamento do projeto.

- **Modelo "Bottom-up" baseado em Atividades**

Esse modelo prevê os procedimentos de estimativa, a partir dos componentes do sistema de software, separadamente, e posteriormente, combinando seus resultados para estimar o projeto como um todo.

As dificuldades na aplicação desse método ocorrem nas primeiras etapas do processo de desenvolvimento de software, quando nem todas as atividades estão identificadas, podendo implicar em erros na estimativa de esforços. Por exemplo, ao basear-se nas atividades identificadas na Estrutura Analítica de Projeto (EAP), estas podem não estar suficientemente detalhadas para serem aplicados os cálculos para estimar os esforços. E uma vez que se obtém tais informações, com o nível de granularidade desejado, a conclusão para uma estimativa final depende da integração desses componentes, não representando exatamente a soma total das estimativas, pois devem ser considerados os esforços para a integração dessas atividades.

A aplicação desse método torna-se difícil quando os recursos de um projeto e seus prazos são escassos, devido ao grande esforço de detalhamento que é exigido.

- **Modelo "Top-down" baseado em Atividades**

O método "top-down", por sua vez, considera as características gerais do projeto. Primeiramente, divide-se o sistema de software em subsistemas, determinando atividades menores e, portanto, mais fáceis de se estimar. O número de divisões deve atingir um nível possível de granularidade, onde se podem estimar os tempos e custos com um certo grau de segurança. A utilização deste método é

mais freqüente nas etapas iniciais do ciclo de desenvolvimento, uma vez que, as características e fatores globais são conhecidos.

Atividades como integração, documentação, gestão e controle do projeto e configuração exigem menos detalhes para serem definidos, e por isso, não necessitam ser subdivididos em sub-atividades para que uma estimativa de esforço e custo seja definida.

No entanto, há probabilidade dos resultados serem pouco realistas, pois uma vez que não o projeto não é detalhado, podem-se obter esforços bem diferenciados quando não são identificadas as complexidades para serem construídos. Estas dificuldades podem, ainda, perdurar da concepção até a construção e implantação do sistema.

Os modelos, apresentados a seguir, aplicam uma ou mais abordagens das anteriormente referenciadas. Boehm (1981), Humphrey (1989) e Kulik (2000) enfatizam a utilização de mais de uma abordagem para obtenção de bons resultados.

2.4 Linhas de código-fonte

A utilização de linhas de código-fonte como modelo de estimativa pode ser facilmente adotada uma vez que representa uma medida direta do tamanho do projeto. Para isso, a formação de um histórico baseia-se em contá-las após as fases desejadas do ciclo de vida do desenvolvimento e armazená-las em banco de dados ou planilhas.

As comparações entre projetos devem considerar as semelhanças de projetos anteriores, como linguagem utilizada. Ainda se observarão diferenças quando há reaproveitamento de código, uso de componentes de software comercialmente adquiridos (COTS) ou utilização de ferramentas geradoras de código, e, neste caso,

podem ser contadas apenas as linhas de código para adaptar o componente ao novo projeto.

Quando não existem disponíveis dados históricos, as alternativas para este método são:

- A utilização de métodos de analogia;
- A utilização do julgamento de especialistas.
- Regressão linear para a aproximação de valores

Os dados são armazenados em milhares de linhas de código-fonte (KSLOC) e podem ser consideradas as linhas em branco, as linhas de comentários, as de declaração de variáveis ou apenas as executáveis. Vijayakumar (1997) fez um estudo com mais de 250 projetos onde eram consideradas opções, apresentadas em um questionário dirigido às prestadoras de serviço do Ministério de Defesa dos Estados Unidos da América. Esse questionário visou à formação de um histórico, além de selecionar fornecedores e medir a qualidade de seus produtos.

Pressman (2000) cita que esse modelo não considera o número de linhas de código utilizadas para se construir unidades de testes, a menos que sejam desenvolvidas em um processo de desenvolvimento de software, detalhado com revisões, planejamentos e documentações.

As maiores dificuldades em se utilizar esse modelo surgiu com as aplicações visuais ou baseadas em janelas (VIJAYAKUMAR, 1997). Por não haver meios de comparação específicos, é aconselhado adotar outros modelos de apoio.

2.5 Pontos de Função

O modelo baseado em pontos de função foi primeiramente apresentado em 1979 por Allan Albrecht (FILGUEIRAS, 2003) apud (ALBRECHT, 1979) e fundamenta-se em analisar o tamanho de um projeto de desenvolvimento de software sem que a etapa de projeto tenha sido realizada.

Utiliza os requisitos dos usuários para estimar o número de linhas de código, o custo e o tempo de construção através de um número gerado a partir da quantidade e, do nível de complexidade das entradas e saídas desejadas, bem como as informações persistidas.

A proposta do modelo se resume em quantificar as capacidades e funções do futuro produto, independentemente de tecnologia ou de linguagem de programação utilizadas.

Os passos básicos para utilização identificados por Braude (2001) são:

1) Identificar as funções que o sistema deve conter, considerando somente a funcionalidade vista pelo usuário e não detalhadamente como em um programa.
2) Para cada uma destas funções, identificá-las conforme uma das funções lógicas:
    a. Entrada Externa (EE)
    b. Saída Externa (SE)
    c. Consulta Externa (CE)
    d. Arquivo Lógico Interno (ALI)
    e. Arquivo Lógico Externo (ALE)
3) Obter o número de pontos de função multiplicando-se o número de cada uma das funções lógicas acima pelo seu fator de complexidade conforme a Figura 2.1.

TIPO DE FUNÇÃO	COMPLEXIDADE FUNCIONAL	TOTAL COMPLEX.	TOTAL TIPO FUNÇÃO
ARQUIVO	SIMPLES X 7 = MÉDIA X 10 = COMPLEXA X 15 =		
INTERFACE	SIMPLES X 5 = MÉDIA X 7 = COMPLEXA X 10 =		
ENTRADA	SIMPLES X 3 = MÉDIA X 4 = COMPLEXA X 6 =		
SAÍDA	SIMPLES X 4 = MÉDIA X 5 = COMPLEXA X 7 =		
CONSULTA	SIMPLES X 3 = MÉDIA X 4 = COMPLEXA X 6 =		
* * * TOTAL DE PONTOS DE FUNÇÃO NÃO - AJUSTADOS =			

Figura 2.1 - Contagem de Pontos de Função não ajustados

4) Encontrar os fatores de ajuste para as 14 características gerais do projeto, demonstradas na Tabela 2.1, e posteriormente, atribuindo seu peso de zero a cinco conforme seu grau de influência, como demonstrado na Tabela 2.2.

Características Gerais do Projeto	
1	Necessidade de restauração e backup?
2	Comunicação de Dados é necessária?
3	Funções para processamentos distribuídos são necessários?
4	O desempenho é um fator crítico?
5	Execução em um ambiente já bem utilizado?
6	Necessita de entrada de dados on-line ou tempo real?
7	Múltiplas telas para entrada de dados?
8	Há campos que devem ser mantidos atualizados on-line ou tempo real?
9	Entradas, saídas e consultas a arquivos são complexas?
10	Processamentos internos são complexos?
11	Código é voltado para reutilização?
12	Conversão e instalação devem estar inclusas?
13	Instalações múltiplas para diferentes organizações?
14	Deve facilitar mudanças e utilização pelo usuário?

Tabela 2.1 - Características Gerais do Projeto - fatores de ajustes para pontos de função

Peso	Grau de Influência
0	Nenhuma
1	Eventual
2	Moderado
3	Necessário
4	Significativo
5	Essencial

Tabela 2.2 - Pesos por Grau de Influência das Características Gerais do Projeto

5) Obter o número de pontos de função ajustado conforme a fórmula abaixo:

$$PF = PF_{NA} \times [0{,}65 + 0{,}01 \times \sum peso \times GI] \quad (2.4)$$

onde:

- PF é o número de pontos de função ajustados;
- PF_{NA} é o número de pontos de função não ajustados;
- GI é o grau de influência.

A conversão de pontos de função para linhas de código pode ser feita com o auxílio da Tabela 2.3 para cada linguagem de programação.

Linguagem de Programação	Média LOC/FP
Assembler	320
C	128
COBOL	106
FORTRAN	106
Pascal	90
C++	64
Ada95	53
Visual Basic	32
Smalltalk	22
Power Builder	16
SQL	12

Tabela 2.3 - Relação LOC/PF por Linguagem de Programação

O fluxo do processo de estimativa, utilizando pontos de função, pode ser resumido conceitualmente conforme mostra a Figura 2.2 (GALVÃO, 1999). Após o planejamento do sistema, com o cálculo dos pontos de função ajustados, obtém-se a estimativa de esforço necessário, utilizando-se as informações sobre a produtividade da equipe. As informações sobre produtividade consideram a experiência do programador e a linguagem de programação utilizada.

Figura 2.2 - Fluxo do Esquema Conceitual da Análise por Pontos de Função

2.6 COCOMO

O COCOMO é um modelo de estimativa de esforço e custo desenvolvido por Barry Boehm e publicado em 1981 (BOEHM,1981) e cuja utilização se faz basicamente através de dados históricos. Ele publicou novas abordagens, entre 1996 e 2000 (BOEHM, 2000), culminando no COCOMO II.

No livro de sua obra (BOEHM, 2000), é fornecida uma ferramenta onde podem ser inseridas informações históricas de projetos, e apontados os direcionadores de custo ("cost-drivers"), que representam fatores de ajuste da produtividade do processo.

Boehm (2000) faz uma relação específica entre o tamanho do projeto, expresso no número de linhas de código-fonte de um software e a quantidade de esforço para construí-lo, contando incluindo os fatores que mudam o comportamento dessa relação por afetarem a produtividade.

O modelo COCOMO representa uma hierarquia de modelos de estimativa que podem ser utilizados de acordo com a qualidade e exatidão necessárias para uma determinada aplicação. Três níveis de detalhamento podem ser utilizados (BOEHM, 2000):

- **Básico**: é realizada uma estimativa rápida sem considerar fatores de ajuste.
- **Intermediário**: são utilizados 15 fatores de ajuste (custo), proporcionando recursos para medir a influência do ambiente na produtividade.
- **Detalhado**: são consideradas todas as características para construir uma estimativa de desenvolvimento de software, desde um módulo até os níveis mais altos, tais como, os subsistemas e o sistema como um todo.

Boehm (2000) faz ainda uma análise do mercado de software, dividindo-o em setores:

- **Usuários finais** – utilizam software para suprir sua necessidade local de obter informações através de geradores de aplicação, planilhas e outros.
- **Infra-estrutura** – provê produtos essenciais como sistemas operacionais, gerenciadores de bancos de dados, gerenciadores de interface gráfica, sistemas de redes.
- **Intermediários** – representados pelos geradores de aplicações, desenvolvedores de aplicações e integradores de sistemas.

O COCOMO II suporta uma família de modelos, personalizada e mais adequada aos setores mencionados acima, de acordo com a sua relação com o mercado de software. Essa adequação se dá, porque as empresas desses setores necessitam de modelos de estimativa que as auxiliem em momentos distintos do projeto de acordo com seu modelo de negócios.

Uma análise de setor intermediário quanto à necessidade de adoção de um modelo de estimativa pode ser descrita como:

- **Composição da Aplicação** – é caracterizada por organizações que vendem componentes de software ou aplicações voltados para a composição de outras aplicações. Sua participação é mais ativa nas primeiras etapas do processo de desenvolvimento de software, na prototipação de interfaces de usuário, mensuração de requisitos de desempenho e na integração de aplicações. Neste caso, o modelo é baseado em pontos de objetos. Esta grandeza é contabilizada com a contagem do número de telas, relatórios, módulos 3GL utilizados em uma aplicação, em que para cada um é atribuído um nível de complexidade (simples, médio, complexo).
- **Geradores de Aplicações, Integradores de Sistemas e Fabricantes de Infra-estrutura** – é utilizada a composição de modelos derivados da necessidade de utilizar fatores de medida específicos de seu produto ou processo, como a utilização de componentes (COTS), a disponibilidade de

software reutilizável, a profundidade de entendimentos dos requisitos, a arquitetura.

A precisão das estimativas é consistente com o nível de conhecimento das informações obtidas até o momento. Por isso, em estágios iniciais do processo de desenvolvimento de um produto, o desvio da estimativa do tamanho de um projeto chega a quatro vezes seu tamanho obtido ao final do processo (BOEHM,2000).

Na adequação dos modelos de estimativa à estratégia do processo utilizado, podem ser conduzidas etapas, conforme demonstra a Figura 2.3 (BOEHM, 2000), para obter estimativas mais exatas ao definir esses momentos de verificação das métricas aplicadas. São elas:

- **Etapa pré-protótipo** – para prototipagens antecipadas de interfaces, interações entre sistemas, questões de desempenho e outros aspectos de alto risco, ocorrendo nas primeiras fases do ciclo de vida.
- **Etapa inicial do projeto** – quando os requisitos já estão bem definidos e uma arquitetura básica do software já foi concebida.
- **Etapa posterior à definição da arquitetura** – durante a fase de construção em que a arquitetura do software, o plano de projeto e a determinação dos riscos já foram bem definidos.

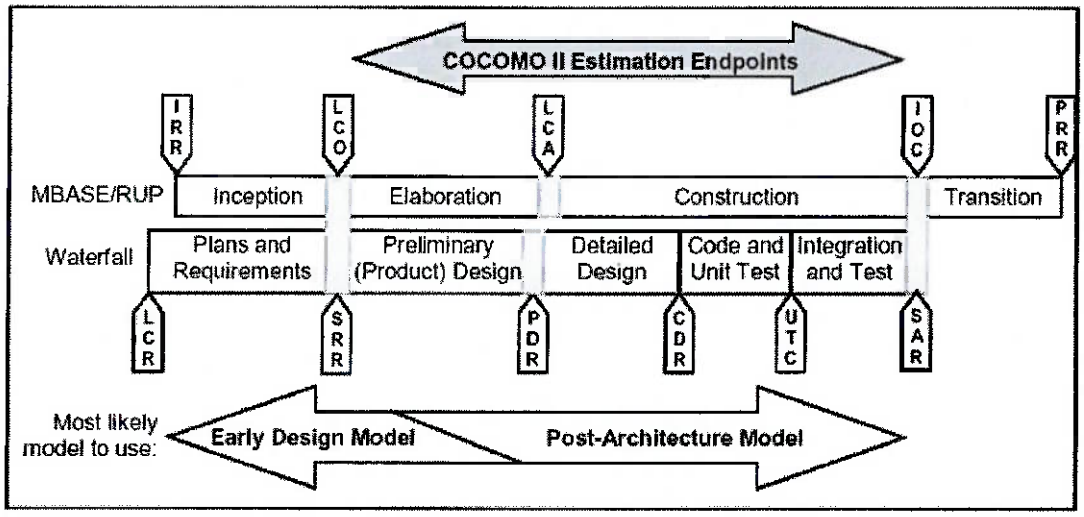

Figura 2.3 – A aplicação do COCOMO nas fases do ciclo de vida

O COCOMO II requer informações sobre o tamanho do projeto para o cálculo das estimativas de esforço. Grandezas como pontos de função ou número de linhas de código são comumente utilizados para o cálculo da estimativa de esforço e do tempo de desenvolvimento necessários. As equações (2.1) e (2.2) descrevem esses cálculos respectivamente para o esforço-mês (PM) e tempo de desenvolvimento (TDEV) (BOEHM, 2000). Ambos os valores carregam o índice NS ("Nominal Schedule") significando que os cálculos não consideram efeitos quanto às mudanças de esforço ou tempo de desenvolvimento.

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i \tag{2.1}$$

$$\text{onde } E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

$$TDEV_{NS} = C \times (PM_{NS})^{F} \tag{2.2}$$

$$\text{onde } F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SF_j = D + 0.2 \times (E - B)$$

O valor de n é igual a 16 para os fatores multiplicadores de esforço (EM), no modelo quando utilizado em uma fase após a definição da arquitetura do software, e igual a 6 para o modelo baseado na fase inicial do processo de desenvolvimento. As constantes A, B, C e D foram obtidas através da calibragem dos parâmetros baseados em 161 projetos catalogados e estudados pelo COCOMO II. Seus valores correntes (BOEHM, 2000) são:

- A = 2.94
- B = 0.91
- C = 3.67
- D = 0.28

O somatório dos 5 valores, fatores de escala (SF), afetam exponencialmente o cálculo do esforço e da produtividade. Seus valores também foram tabulados na primeira versão do COCOMO em 1981 (BOEHM, 1981) e posteriormente calibrados

para atender às mudanças emergentes dos processos de desenvolvimento de software, aos novos critérios para mensuração do tamanho de programas e ao aumento da comercialização de software adquiridos comercialmente (COTS).

Valores são atribuídos a esses fatores escala, através da calibração dos parâmetros dos 161 projetos estudados. Essa atribuição é realizada com base em faixas de influência:

1. Muito baixa;
2. Baixa;
3. Média ou neutra;
4. Alta;
5. Muito alta;
6. Extra alta.

Os cinco fatores de escala são:

- **Precedência (PREC)** representa a quantidade de projetos ou produtos já desenvolvidos similares ao produto em questão.
- **Flexibilidade de Desenvolvimento (FLEX)** em que a conformidade com requisitos pré-estabelecidos ou com as especificações de interfaces externas promove uma baixa flexibilidade para o desenvolvimento.
- **Resolução de Riscos (RESL)** refere-se à capacidade de avaliar e mitigar os riscos inerentes ao processo de desenvolvimento e à arquitetura.
- **Coesão da Equipe de Desenvolvimento (TEAM)** expressa a experiência de trabalho em grupos e o grau de familiaridade da equipe designada com o ambiente e a cultura do cliente, favorecendo a comunicação para atender adequadamente os objetivos do cliente.
- **Maturidade do Processo (PMAT)** classifica a organização nos níveis de maturidade do CMM do SEI, Instituto de Engenharia de Software. Essa classificação é obtida com a validação da aderência do processo de desenvolvimento às áreas de processo-chave (KPA).

No entanto, o COCOMO ainda utiliza os fatores multiplicadores de esforço ou direcionadores de custo, que na versão COCOMO II, inclui 17 variáveis para o modelo posterior à definição da arquitetura e 7 para o modelo utilizado no início do projeto. Esses direcionadores de custo são classificados da seguinte maneira:

- **Fatores relacionados ao Produto**
  - **Confiabilidade Requerida do Software (RELY)** reflete o impacto causado por uma falha para um programa que precisa desempenhar uma função em determinado período de tempo.
  - **Tamanho da Base de Dados (DATA)** reflete o efeito que testes com grandes quantidades de dados tem sobre o desenvolvimento do produto.
  - **Complexidade do Produto (CPLX)** reflete a combinação de áreas como operações de controle, operações computacionais, operações dependentes de dispositivo específico, operações de administração de dados, e operações de gestão de interface homem-computador.
  - **Desenvolvimento voltado para Reuso (RUSE)** reflete o impacto de um desenvolvimento mais genérico para suportar o reuso em projetos futuros.
  - **Documentação se aplica às Necessidades do ciclo de vida (DOCU)** mede a qualidade da documentação exigida quanto à sua aderência ao ciclo de vida adotado.
- **Fatores relacionados à Plataforma**
  - **Restrições quanto ao Tempo de Execução (TIME)** que podem ser medidos pela razão porcentual do tempo de execução pelo total disponível.
  - **Restrições quanto ao Meio de Armazenamento (STOR)** reflete a quantidade porcentual de uso dos meios de armazenamentos disponíveis.
  - **Volatilidade da Plataforma (PVOL)** reflete a freqüência de mudanças nas plataformas utilizadas pelo produto.

- **Fatores relacionados a Pessoas**
  - **Capacidade do Analista (ACAP).**
  - **Capacidade do Programador (PCAP).**
  - **Continuidade da Equipe (PCON)** reflete o grau de rotatividade da equipe.
  - **Experiência com o Tipo de Aplicação (APEX).**
  - **Experiência com a Plataforma (PLEX).**
  - **Experiência com as Ferramentas e Linguagens Utilizadas (LTEX).**
- **Fatores relacionados ao Projeto**
  - **Uso de Ferramentas de Software (TOOL)** reflete a influência das ferramentas utilizadas na produtividade da equipe. Esse fator destacou-se muito, e por isso, originou a extensão do COCOMO, o CORADMO (BOEHM, 2000).
  - **Desenvolvimento em vários pólos (SITE)** reflete a influência da produtividade em equipes trabalhando separadamente quanto à comunicação, suporte, gestão de configuração, padronização e integração.
  - **Cronograma de Desenvolvimento Requerido (SCED)** reflete as necessidades de se encurtar o cronograma.

Quando o modelo é utilizado no início do projeto, em que ainda não se tem uma arquitetura do software bem definida e os requisitos não estão completamente definidos, os fatores direcionadores de custo que devem ser utilizados são:

- Confiabilidade e Complexidade do Produto **(RCPX).**
- Desenvolvimento voltado para Reuso **(RUSE).**
- Dificuldades com a Plataforma **(PDIF).**
- Capacidade da Equipe **(PERS).**
- Facilidades **(FCIL).**
- Cronograma de desenvolvimento requerido **(SCED).**

Hale et al. (2000) estudaram outros fatores que poderiam melhorar a precisão das estimativas. O trabalho envolveu a utilização de métricas na designação de tarefas, então apresentaram novos fatores de escala, que contribuíram para a utilização do COCOMO. Esses parâmetros são:

- Intensidade;
- Concorrência;
- Fragmentação.

Uma tarefa é definida por Hale et al. (2000) como uma das atividades específicas no processo de desenvolvimento de software, o desenvolvimento de um módulo, o desenvolvimento de uma aplicação ou de outra atividade relacionada.

Hale et al.(2000) concluiu alguns resultados com os novos fatores de escala:

- **Intensidade:** mede o grau de enfoque dedicado a uma determinada tarefa e sua continuidade. Tarefas com baixa intensidade caracterizam-se por freqüentemente serem interrompidas ou apresentarem longos períodos descontínuos de dedicação.
- **Concorrência:** expressa o grau de cooperação e colaboração entre pessoas e equipes que trabalham em tarefas comuns ou relacionadas.
- **Fragmentação:** refere-se à divisão do tempo de uma equipe em múltiplas tarefas.

Para medir esses fatores, Hale et al. (2000) considerou que uma unidade de tempo é descrita por um intervalo de tempo de duração fixa, enquanto a duração da tarefa é relacionada com o número de unidades de tempo entre seu início e conclusão. Os impactos das medidas associadas à designação de tarefas concluem que:

1. O esforço de desenvolvimento diminui com o aumento da intensidade. Cada interrupção causa um tempo necessário para que se recupere o enfoque da tarefa.
2. O esforço de desenvolvimento aumenta com a concorrência. Um desempenho melhor é observado em tarefas, em que é possível que, os desenvolvedores

trabalhem de forma mais independente. Há uma tendência natural para que esses profissionais prefiram trabalhar sozinhos.

3. O esforço de desenvolvimento aumenta com a fragmentação. Quando o tempo é muito dividido entre tarefas, a mudança de uma tarefa para outra pode consumir um tempo considerável.

O ajuste desses fatores proposto por Hale et al. (2000), é apresentado conforme a Tabela 2.4.

Métrica	Muito baixo	Típico	Muito alto
Intensidade	1.3	1	0.7
Concorrência	0.9	1	1.3
Fragmentação	0.9	1	1.4

Tabela 2.4 - Ajuste de Fatores das medidas associadas à designação de tarefas.

2.7 Comparação entre os Modelos

Os modelos de estimativas de esforço, apresentados neste capítulo, têm sido amplamente utilizados pelas organizações. A escolha em se adotar um dos modelos, geralmente, parte de profissionais que já tenham algum contato ou através de fornecedores ou clientes que já o adotam.

Ambos os modelos necessitam de dados históricos visando melhorar as estimativas. Além disso, também precisam ser calibrados para as condições da organização. Para o COCOMO, o ajuste é realizado ajustando-se os valores dos fatores de escala e dos multiplicadores de esforço, através da ferramenta fornecida no livro COCOMO II (BOEHM, 2000). Na análise por Pontos de Função, as constantes de complexidade também podem ser alteradas, bem como, os pesos dos graus de influência das características globais do projeto. O momento para a realização dessa calibração no COCOMO é nas etapas descritas neste capítulo, enquanto que, em Pontos de Função, não há uma regra, mas geralmente é realizada ao final do projeto.

A análise por Pontos de Função propõe-se a fornecer informações do tamanho do projeto, independente da linguagem de programação ou da tecnologia, mas também não fazem restrições quanto ao processo de desenvolvimento, nem à capacidade dos profissionais envolvidos. Esses parâmetros podem ser utilizados no cálculo da produtividade da organização. Ao passo que, o COCOMO II disponibiliza esses parâmetros para ajustar o modelo e é sensível quanto à maturidade do processo de desenvolvimento utilizado pela organização. O COCOMO fundamenta-se no tamanho do projeto. Esse tamanho pode ser calculado pelo número de linhas de código ou utilizando pontos de função. É possível fazer a conversão entre essas grandezas utilizando Tabela 2.3.

O COCOMO disponibiliza um conjunto de parâmetros pré-ajustados aos 161 projetos estudados. A sua utilização é dificultada pelo número de parâmetros que devem ser considerados. Por isso, requer uma equipe especializada no modelo, que deve realizar a coleta de dados, analisá-los e deve calibrar o modelo freqüentemente. No caso Pontos de Função, esse papel pode ser atribuído inicialmente ao gerente de projeto ou a um analista. Sua adoção pode ser mais facilitada, por exigir menos formalidades no processo de coleta de dados.

Contudo, a transição de um modelo para outro não é muito fácil. Apesar do COCOMO requerer o tamanho do projeto, deve exigir mais informações do histórico de projetos para fornecer estimativas de esforço e tempo de desenvolvimento. Enquanto que, migrar do COCOMO para Pontos de Função é possível.

CAPÍTULO 3

3 HISTÓRICO DE PROJETOS

3.1 Introdução

As informações armazenadas em um histórico de projetos são quantitativas e relacionadas ao tamanho de parte ou de todo um produto desenvolvido mensurado em número de linhas de código, ao esforço em homens-mês e ao custo. Portanto, quanto mais detalhado for esse histórico mais exata é a estimativa.

No caso do modelo baseado em Pontos de Função, o histórico de projetos também se destina a:

- Monitorar os indicadores de produtividade baseadas em pontos de função;
- Viabilizar a avaliação de índices de qualidade nos projetos em função de seu tamanho medido em pontos de função;
- Facilitar a análise de custos do projeto referentes ao total de pontos de função;
- Fornecer comparação entre projetos a partir de critérios de similaridade.

No entanto, ainda pode ser observado que gerentes de projeto e até mesmo os programadores ficam relutantes em cumprir um programa, por julgarem perda de tempo, temerosos quanto à possibilidade de estarem sendo medidos quanto à capacidade técnica. Kulik (2000) afirma ser difícil obter um apoio imediato da alta gerência, pois, um programa de métricas sem objetivos voltados ao negócio da organização, não justifica um investimento.

Em 1982, pesquisas foram realizadas por um laboratório de engenharia de software na NASA. Após vários anos, realizando pesquisas com o processo de coleta de dados voltados para a formação de históricos de projetos, demonstraram que coletar dados é um processo de alto custo. Concluíram, também, que esse processo consome cerca de

15% dos custos de desenvolvimento, considerando a utilização de mais de 700 grandezas (HUMPHREY, 1989). Esses resultados foram explicados por:

- Inexperiência das pessoas envolvidas em coletar esses dados;
- Utilização de um procedimento manual para coleta dos dados;
- Ausência de recursos computacionais para os cálculos de análises e de validações desses dados.

A Tabela 3.2 (HUMPHREY, 1989) mostra a perda porcentual de produtividade em cada atividade executada durante o estudo.

Atividade	**Perda de produtividade obtida**
Formulários, reuniões, treinamentos, entrevistas, custo de utilização das ferramentas.	3 a 7%
Processamento dos Dados: Coletando e validando os dados; Armazenando os dados; Extraindo relatórios.	10 a 12%
Análise das informações:	Até 25%

Tabela 3.1 - Custo do Processo de Coleta de Dados para a Formação de Histórico

3.2 Criação e Formação

A criação de um histórico de projetos é essencial para a utilização de um modelo de estimativa (HUMPHREY, 1989), (BOEHM, 2000) e (PRESSMAN, 2000). Uma vez que os modelos, tais como, o COCOMO e o modelo baseado em Pontos de Função, são modelos de estimativa paramétricos. Portanto, a contribuição de um histórico se torna mais necessária, à medida que, os parâmetros significam a base de cálculo para a calibração do modelo.

Humphrey (1989) apresenta um processo para criação e formação de um histórico de projetos. Este processo pressupõe a elaboração de um plano para implantação de um programa de métricas na organização. Além disso, ele sugere métodos de validação dos dados coletados. Este plano deve estabelecer critérios para:

1. Escolher adequadamente os dados necessários e definir quem os utilizará e quais são os objetivos da organização.
2. Definir a especificação dos dados a serem coletados.
3. Identificar as pessoas envolvidas na coleta dos dados, bem como esclarecer seu comprometimento em fazê-lo.
4. Fornecer suporte para esse processo.
5. Definir como os dados devem ser coletados, além de fornecer uma documentação.
6. Elaborar um procedimento para validação dos dados.
7. Definir um processo para gerenciar esses dados.

Kulik (2000) sugere que a seleção das métricas pode ser iniciada aplicando-se algumas medidas básicas, tais como, o número de linhas de código e o tempo utilizado para a conclusão de atividades. Mesmo assim, a formação de um histórico de projetos demanda um investimento da organização, por isso, Kulik (2000) defende que para a obtenção de apoio da alta gerência, a adoção de um plano de métricas deve ter o foco no negócio da organização.

A utilização do modelo baseado em pontos de função deve reunir informações para a formação do histórico de projetos, seja na etapa de cálculo ou no ajuste dos pontos de função, geralmente ao final do projeto, tais como:

- Dados do Projeto;
- Informações básicas sobre a plataforma utilizada;
- Tipo de linguagem utilizada;
- Estágio de desenvolvimento;
- Presença de processamento distribuído no sistema;
- Fatores de Ajuste no Cálculo de Pontos de Função;

- Produtividade medida.

A Tabela 3.2 exemplifica algumas informações que podem ser obtidas para a formação do histórico da organização.

Descrição	**Unidade**
Equipe	Homem-mês
Duração	mês
Linhas de documentação	Linhas
Linhas de Código	LOC
Custo	US$, por exemplo.

Tabela 3.2 - Dados a serem obtidos no processo de formação de histórico

Outras informações também podem fornecer subsídios na análise dos dados, como:

- Os modelos de desenvolvimento utilizados;
- A situação política e econômica na época do projeto;
- Identificação das pessoas responsáveis pela coleta dessas informações.

No COCOMO e no COCOMO II, a formação do histórico de projetos pode ser realizada através dos formulários disponíveis nas referências sobre o modelo (BOEHM, 2000). Esses formulários de coleta de dados são utilizados, conforme a etapa de desenvolvimento, pré-projeto ou pós a definição da arquitetura do software. Cada etapa requer diferentes tipos de informação para se efetuar estimativas de esforço, por isso, os direcionadores de custo utilizados em cada uma das etapas são diferentes, de acordo com as informações que já são conhecidas. O COCOMO II expandiu sua abordagem em relação aos processos de desenvolvimento suportados. Na primeira versão, em 1981, apenas o processo em cascata era considerado pelos direcionadores de custo, mas em 2000, passou a dar suporte aos processos de desenvolvimento de software, tais como, o processo incremental (PRESSMAN, 2001) e o processo unificado (JACOBSON, 1999). Os conceitos de maturidade de

software também são suportados pelo COCOMO II, conforme os padrões do CMM[1] (HUMPHREY, 1989) e suas áreas de processo-chave.

De uma forma geral, os dados para formação de um histórico, não devem ser complexos ou consumir muito tempo para serem coletados. Portanto, para que a análise seja simples, os dados devem ser verificáveis e consistentes.

3.3 Interpretação e utilização

A interpretação dos dados coletados, para formação de um histórico de projetos, fornece condições para a organização melhorar seu processo de desenvolvimento ou a qualidade de seu produto, à medida que, utiliza as medidas diretas e indiretas que favorecem o processo de gerenciamento do custo do projeto. Nesse processo (PMBOK, 2000) incluem atividades, como:

- Planejar melhor os recursos;
- Estimar melhor os custos;
- Alocar os custos nas atividades do projeto;
- Controlar melhor os custos.

Algumas medidas podem ser obtidas a partir do tamanho em número de linhas de código, como mostra a Tabela 3.3 (FILGUEIRAS, 2003) e sua forma de cálculo:

Descrição da medida	**Unidade**
Produtividade	KLOC / HH
Qualidade	Defeitos / KLOC
Custo	$ / KLOC
Documentação	$ / KLOC
Portfolio	KLOC total da empresa
Manutenção	Tempo / KLOC

[1] CMM - Capability Maturity Model é um modelo de melhoria contínua do processo de desenvolvimento de software desenvolvido pelo SEI. O CMM tem como objetivo melhorar o processo através da resolução das questões principais em cada um dos cinco níveis de maturidade.

Tabela 3.3 - Medidas derivadas do Tamanho do Código

Para o COCOMO (BOEHM, 2000), a padronização da contagem do número de linhas de código, por exemplo, exige um detalhamento quanto às considerações específicas como, por exemplo, a inclusão de linhas de código de comentário. Eventuais diferenças neste aspecto afetam diretamente a comparação com projetos anteriores.

Os dados coletados no processo definido devem ser considerados, se e somente se, são comparáveis (BIELAK, 2000; MAXWELL, 2001). Maxwell (2001) define que os dados são comparáveis quando são expressos nas mesmas unidades, se fazem parte da mesma organização em contextos semelhantes; e quando são medidos da mesma forma. O COCOMO utiliza uma base de dados de 161 projetos, fornecidos pelos afiliados, favorecendo esta comparabilidade. As diferenças são parametrizadas, de forma a permitir a equalização de dados não diretamente comparáveis.

No modelo de estimativa baseado em pontos de função, a atualização do histórico pode ser realizada ao término de cada uma das fases do projeto, ou ao término de cada etapa realizada. Como não há parâmetros específicos para cada etapa, então, a utilização desses dados, no mesmo projeto, pode melhorar as estimativas. Comparar e analisar as produtividades esperada e real das etapas anteriores implica em uma capacidade maior para gerenciar o projeto.

Segundo Bielak (2000), alguns fatores são considerados básicos para a obtenção de uma medida de produtividade que capacitam a análise do histórico de projetos:

- Dados históricos;
- Medidas de complexidade de sistemas;
- Experiência dos profissionais;
- Restrições de projeto;
- Tamanho do projeto.

Alguns fatores de produtividade utilizados nos modelos de estimativa e na formação do histórico de projetos podem ser impactados, como por exemplo, a ausência ou

inconsistência de dados que não foram devidamente coletados, seja, pela falta de rigor no acompanhamento das métricas do projeto ou indefinição dos objetivos da organização. Além disso, os envolvidos no cálculo dos pontos de função podem contribuir para o preenchimento incorreto dos formulários de coleta de dados. Dentre alguns motivos, destacam-se:

1. A falta de tempo;
2. O falta de entendimento do processo de medição;
3. A falta de conhecimento de como coletar os dados ou a falta de informação das unidades (tempo, linhas de código) que devem ser utilizadas;
4. A opção por omitir alguma informação que possa medir diretamente a produtividade do profissional, que acredita estar sendo avaliado, e não que ele faz parte de uma equipe, que por sua vez, segue um processo de desenvolvimento.

Nestes casos, as estimativas para projetos futuros são prejudicadas. Por outro lado, é observado o descarte freqüente dessas informações, provocando a perda de dados que podem representar um impacto nos custos dos respectivos projetos. Em (STRIKE et al., 2001), algumas técnicas são comparadas e simuladas para melhorar a precisão das estimativas de custo quando há dados incompletos no histórico. As técnicas consistem em inserir os dados faltantes utilizando os outros projetos como referência, e obtê-los através do cálculo por regressão linear. Strike et al.(2001) afirma que a utilização de dados de projetos semelhantes também pode ser uma alternativa interessante.

CAPÍTULO 4

4 AVALIAÇÃO DO GRAU DE COMPLEXIDADE DE UM SISTEMA

4.1 Introdução

A complexidade de um sistema está relacionada com a dificuldade em realizar sua construção. Desta forma, é importante obter informações sobre a complexidade qualitativamente e quantitativamente, tendo-se como base os recursos disponíveis, a experiência dos profissionais envolvidos e o histórico de projetos.

O objetivo de quantificar uma grandeza, indicando o grau de complexidade, para a execução de uma tarefa, está relacionado aos ajustes nos cálculos nos modelos de estimativa apresentados no capítulo 2, tais como Pontos de Função e COCOMO. Sua contribuição está no ajuste do esforço necessário para a realização de uma determinada tarefa conforme a complexidade atribuída, de modo a justificar, matematicamente, as diferenças obtidas em tarefas semelhantes.

Alguns autores, como McCabe (1976) e Halstead (1977), dimensionam o nível de complexidade de um programa por um número qualitativo em função do número de linhas do código-fonte, assim como, o número de decisões ou número de operandos e operadores. Ao utilizar um modelo de estimativa de esforço, tais como, pontos de função ou COCOMO, essas considerações podem ser realizadas de maneira menos formal, sem a obrigatoriedade de se efetuar cálculos, através da utilização de tabelas que auxiliam a classificação do grau de complexidade.

4.2 Métrica de Complexidade de McCabe

A complexidade de McCabe foi apresentada em 1976 (MCCABE, 1976). É uma métrica baseada na medida de complexidade lógica de um programa ou de um software que pode ser obtida através do número de caminhos possíveis na execução de um programa. Nesta medida, são utilizados grafos para representar os fluxos de

controle de execução de um programa, denominados de **Complexidade Ciclomática**.

O cálculo da complexidade ciclomática, conforme evidenciado por McCabe (1976), pode ser expresso com a equação (4.1):

$$V(G) = e - n + 2p \tag{4.1}$$

Onde:

V(G) é a complexidade de um grafo G;

e é o número de ramos;

n é o número de vértices;

p é o número de componentes conectados.

Diz-se que um grafo é fortemente conectado, quando V(G) é igual ao número máximo de caminhos linearmente independentes. O cálculo da complexidade ciclomática considera apenas os caminhos a serem seguidos pelo programa (estrutura de decisão), não considerando a quantidade de instruções contidas durante a execução de um ramo.

A utilização desse cálculo, na prática, deve ser feita com o auxílio de uma ferramenta que auxilie na automatização dessa contagem assim como, nos seus cálculos. Em um mesmo vértice, de acordo com McCabe (1976), pode-se encontrar um ou mais ramos que saem de um mesmo vértice.

Na prática, sua aplicação deve ser associada com outro modelo, que considere o número de complexidade como uma métrica direta que pode ser armazenada em histórico. Contudo, Kan (2002) observou que este modelo não tem bons resultados com construções lógicas, como por exemplo, funções recursivas. Em projetos orientados a objetos, a utilização desta técnica deve considerar características, tais como, o polimorfismo e a herança visto que, não se pode analisar apenas a estrutura estática do sistema.

4.3 Equações de Halstead

Halstead (KAN, 2003 apud HALSTEAD, 1977) propôs equações que contabilizam o tamanho de um programa através do cálculo de uma grandeza denominada **Volume**, demonstrada na eq.(4.1) abaixo:

$$V^* = N \times \log_2(n) = (N^*_1 + N^*_2)\log_2(n_1^* + n_2^*) \quad (4.1)$$

onde:

N = número total de ocorrências de operandos e operadores

n = número de operandos e operadores distintos

N_1^* = estimativa do total de ocorrências de operandos

N_2^* = estimativa do total de ocorrências de operadores

n_1^* = estimativa do número de operandos únicos

n_2^* = estimativa do número de operadores únicos

V^* = volume de trabalho envolvido

A utilização das equações de Halstead, requer um processo de identificação dos operandos e operadores. Sua aplicação é mais adequada para programas, cuja linguagem de programação é de baixo nível, como Assembler e C. Nestas linguagens, a sintaxe predominante é efetivamente composta de decisões e operações de execução e desvio. Ao passo que, nas linguagens de mais alto nível, tais como, nas orientadas a objetos ou nas que utilizam componentes. Reifer (2002) sugere um mapeamento dos operadores como objetos e dos operandos como métodos ou serviços. Contudo, apesar das alternativas para a continuação da utilização desta técnica, a mesma se torna inviável sem o uso de ferramentas, que auxiliem na contagem de operadores e operandos.

4.4 Níveis de Complexidade para Pontos de Função

No modelo baseado em pontos de função, a definição de complexidade está relacionada com o tipo de função lógica. Sua influência ocorre no momento da

obtenção do número de pontos de função não-ajustados, após a definição e seleção das funções lógicas do novo projeto.

Os níveis de complexidade considerados são classificados como Baixo, Médio e Complexo para cada função lógica. Com o número total de cada uma, obtém-se o peso conforme o numero de registros ou tipos de arquivos, como é demonstrado na Tabela 4.1 (PRESSMAN, 2002 apud. BOEHM, 1996).

Tipo de Objeto	**Fator de complexidade**		
	Simples	Médio	Complexo
Telas	1	2	3
Relatórios	2	5	8
Componentes 3GL	-	-	10

Tabela 4.1 - Fator de peso de complexidade dos objetos

CAPÍTULO 5

5 DIRETRIZES PARA APLICAÇÃO DO MODELO

5.1 Introdução

A aplicação de um modelo de estimativa deve ser realizada juntamente com um plano de projeto que a conduza e disponha de critérios específicos para a formação de um histórico. Um plano de projeto define o trabalho e como este será realizado. Deve-se definir cada uma das tarefas principais, estimar o tempo e os recursos necessários, além de considerar uma estrutura de trabalho de gestão e de controle (HUMPHREY,1989).

A adoção de um plano de métricas é apoiada pelo insucesso de planos elaborados de maneira incompleta, cujos recursos não foram corretamente dimensionados, e por isso, incorrem em atrasos no cronograma, na perda de qualidade, no aumento dos custos do projeto e no uso inadequado dos recursos.

Essas questões convergem para a necessidade de uma correta estimativa do trabalho a ser executado em termos de tempo e recursos, que por sua vez, são baseados em estimativas sobre o tamanho do projeto e sua complexidade.

5.2 Identificação dos Objetivos da Organização

É necessário que os objetivos da organização sejam bem definidos, quanto à adoção de um programa de métricas, a fim de tornar possível a obtenção dos resultados desejados. A redução de custos ocorre quando o programa de métricas é bem definido e enfocado nos negócios, garantindo a viabilidade de investimentos na área de tecnologia da informação (TI) (KULIK, 2000). Na grande maioria das organizações, o apoio da alta gerência não é imediato, uma vez que um programa de métricas não proporciona, em curto prazo, benefícios para a organização, , especialmente, quando se está iniciando um programa de métricas. Contudo, esse

apoio é concretizado, com mais facilidade, quando os objetivos da implantação do programa são voltados ao negócio da organização.

Segundo Kulik (2000), os objetivos mais comuns encontrados nas organizações são:

- Melhorar a qualidade do software instalado;
- Obter comprometimentos em relação ao projeto;
- Reduzir os custos de suporte pós-instalação;
- Prevenir problemas de cronograma e custos;
- Aumentar o rendimento da equipe e da organização;
- Estimar rapidamente os projetos de software;
- Entregar rapidamente novos produtos.

5.3 Plano de Projeto

O plano de projeto deve ser desenvolvido no início do processo de desenvolvimento, de aquisição ou de manutenção do software, o seu refinamento deve ocorrer durante o percurso do projeto. Os motivos principais para a elaboração de um plano de projeto (HUMPHREY, 1989), podem ser enumerados:

1. Os requisitos muitas vezes vagos e incompletos no início do projeto. O plano tem o papel de concretizá-los em requisitos precisos e exatos.
2. O projeto conceitual é obtido no início do trabalho juntamente com a primeira concepção de todas as atividades a serem executadas, facilitando a estimativa de recursos e tempo.
3. Para cada refinamento do plano, obtêm-se estimativas mais precisas, favorecendo a gestão do projeto.
4. O detalhamento de implementação e as documentações devem ser incorporados ao plano, à medida que o projeto evolui.

O ciclo de planejamento pode ser resumido nas seguintes etapas:

1. Definição dos requisitos de software.
2. Comprometimento com o plano.
3. Detalhamento das tarefas a serem executadas em uma EAP.
4. Dimensionamento dos elementos da EAP quanto ao tempo de execução e recursos necessários.
5. Alocação de recursos.
6. Elaboração do cronograma.

Ao final, não havendo concordância na negociação de prazos e custos, o ciclo deve ser reiniciado. Apesar do aspecto negativo de se reiniciar o processo, esses ciclos devem realimentar o histórico em favor de planos mais bem elaborados.

5.4 Seleção das Métricas

A escolha das métricas a serem coletadas e posteriormente analisadas, deve ser suportada pelos objetivos da organização (KULIK, 2000). As grandezas que são diretamente medidas, tais como, o número de linhas de código-fonte, o tempo utilizado na construção e o número de defeitos, trazem informações necessárias para a obtenção das medidas indiretas, como produtividade por profissional, taxa de entrega e qualidade.

5.5 Coleta de dados para o histórico

Após a definição dos dados que devem ser selecionados, e, com um modelo de estimativa definido, as atividades de coleta de dados podem ser iniciadas em novos projetos, utilizando os recursos e diretrizes fornecidas pelo modelo.

Para os projetos anteriores, alguns dados podem ser reaproveitados, uma vez que, estão disponíveis, tais como:

- Linhas de código, que podem ser medidas novamente;
- Tempo de duração do projeto, podem ser obtidos pela documentação do projeto.
- Número de profissionais envolvidos, também pode ser obtido utilizando a documentação do projeto.

Dentre as informações que não estão mais disponíveis, algumas podem ser obtidas com a consulta das informações dos projetos anteriores, utilizando a aproximação de valores através, por exemplo, de regressão linear, com base em projetos semelhantes. Destacam-se o número de defeitos, o tempo utilizado em suporte após a entrega, o grau de coesão da equipe e o tempo de desenvolvimento e esforço utilizados em cada etapa do projeto.

Essas informações contribuem significativamente nas estimativas realizadas. Logo, ao se fazer aproximações para obter valores inexistentes no histórico de projetos, é importante observar correlações existentes entre os dados dos projetos anteriores, de forma que situações atípicas não reflitam uma média irreal. No entanto, a prática de se aproximar os dados, através de técnicas de regressão, têm-se obtido melhores resultados, se comparado com o simples descarte dos projetos que não dispõem de todas as informações (STRIKE, 2001).

Estas considerações também devem ser feitas, quando existe um histórico de projetos. Segundo Strike (2001), o descarte das informações de projetos, cujos dados são incompletos, pode causar uma perda significativa de informações que compõem a base para o cálculo de estimativas de esforço e recursos.

5.6 Automatização dos procedimentos de Medição

Os modelos de estimativas, apresentados neste trabalho, requerem o uso intenso de recursos computacionais. Especialmente se o projeto for de tamanho médio, podem ser necessárias ferramentas específicas para os cálculos de esforço, tempo de desenvolvimento e produtividade.

5.7 Uso das métricas no apoio à decisão

Uma vez obtido um conjunto de dados, que compõe o histórico de projetos, através do auxílio de ferramentas de análise, tais como planilhas e gráficos, a organização deve analisar os dados visando à melhoria de seu processo ou à resolução do problema identificado no início do processo. A equipe designada para realizar e controlar o programa de métricas, deve encontrar as correlações entre as grandezas medidas a fim de alcançar os objetivos inicialmente estabelecidos.

O programa de métricas, baseado em um modelo de estimativas, pode auxiliar a organização a antever tendências não favoráveis durante o percurso do projeto. Não adotar, disciplinadamente e persistentemente um programa de métricas, favorece problemas no gerenciamento do cronograma, dos recursos e do custo, como mostra a **Figura 5.1.**

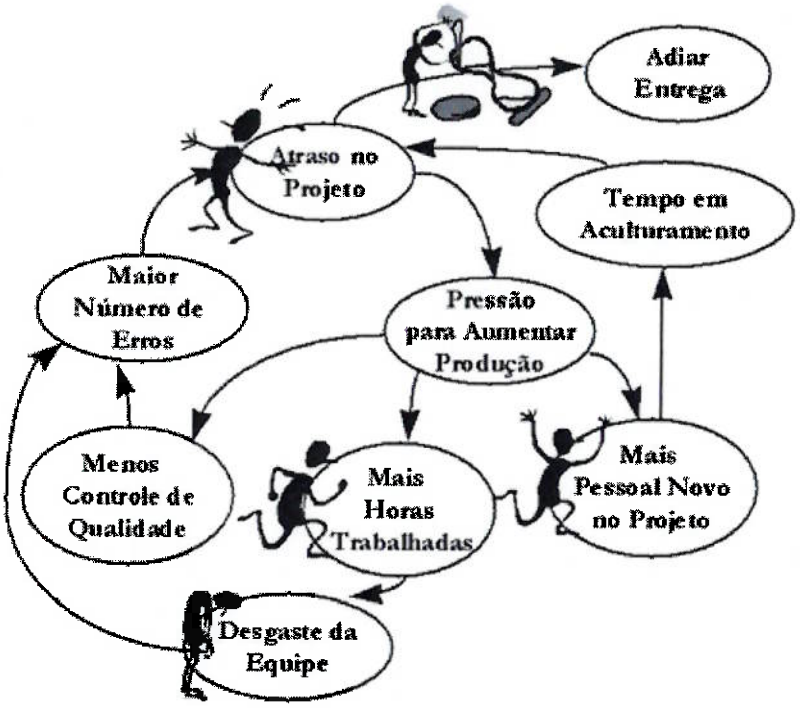

Figura 5.1 - Fluxo de Causas e Efeitos em projetos com estimativas mal calculadas

CAPÍTULO 6

6 APLICAÇÃO EM SISTEMAS BASEADOS EM WEB

6.1 Introdução

A Internet iniciou uma transformação dos negócios trazendo benefícios mesmo após o desaquecimento conhecido como "Bolha da Internet" (TAKATA, 2002), sendo considerada como um modelo de negócios revolucionário em relação aos paradigmas da economia tradicional.

Os sistemas baseados em Web foram responsáveis por despertar nos pesquisadores uma necessidade de analisar, de adequar e de reavaliar os modelos existentes há mais de 10 anos. Os sistemas baseados em Web, sempre que comparados, mostravam-se menos disciplinados, em relação ao processo de desenvolvimento e mais rápidos quanto a entrega dos executáveis (REIFER, 2002) e (MENDES, 2001). A partir de 2000, alguns pesquisadores começaram a apresentar suas propostas para adequar os sólidos e estáveis modelos de estimativas para esses sistemas, ainda com poucos anos de existência.

6.2 Transformação dos Negócios

Os sistemas baseados em Web tornaram-se expressivos por volta de 1997, impulsionados pelo aumento da utilização da Internet de maneira mais interativa, através de aplicações que pudessem, além de levar informações institucionais e de produtos à rede, proporcionar uma personalização desta informação.

Posteriormente, a partir de 1998, iniciou-se um período de utilização da Internet para efetuar transações, onde se aplicam os sistemas voltados para comércio eletrônico, aplicações de apoio à gestão da cadeia de suprimentos, B2B com suporte a EDI, B2C e interfaces para outros sistemas, como os sistemas de gestão empresarial (ERP).

Para a construção de sistemas deste tipo devem ser considerados outros elementos, uma vez que diferem dos sistemas tradicionais. Desta forma, os modelos de estimativas devem ser adaptados ou calibrados a fim de se obter informações de custo e esforço compatíveis.

As tecnologias oferecidas no início da Internet eram baseadas no protocolo de rede TCP/IP, especialmente o protocolo HTTP para transferência de páginas web dos servidores aos programas navegadores ("browsers"), utilizando uma linguagem de marcação, o HTML, que foi derivada do SGML. Essas tecnologias se consolidaram em favor de um padrão aberto.

Em um primeiro momento, a velocidade de transmissão das informações contidas em páginas Web era muito baixa, portanto, sua publicação exigia a otimização dos tamanhos das páginas e imagens trafegadas na rede. Ainda hoje, para páginas e aplicações baseadas em Web, com grande visitação, é indispensável minimizar o tráfego em rede, evitando o alto custo e dando mais conforto para quem está acessando as informações.

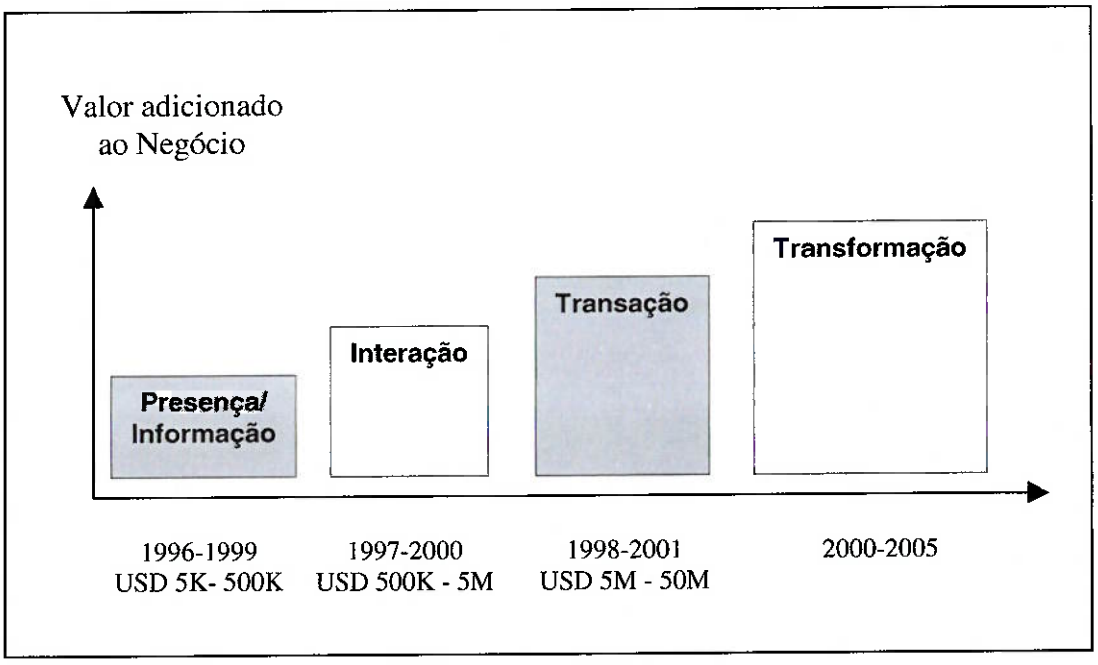

Figura 6.1 - Períodos de Transformação dos Negócios realizados através da Internet

Os sistemas baseados em Web foram inicialmente construídos como programas de comunicação com outras as aplicações (CGI). Posteriormente foram embutidas no servidor de páginas visando melhor desempenho, e atualmente utilizam-se servidores de aplicação em arquitetura distribuída. O retorno da informação solicitada, que anteriormente se dava apenas por HTML se estendeu, culminando no formato XML.

Em ambas, temos tipicamente sistemas identificados como arquitetura cliente-servidor (KAZUO, 2002), porém pouco processamento é realizado no cliente. Portanto, cabe às aplicações gerar os resultados nestes formatos reconhecidos.

No entanto, aplicações Web não são somente compostas de HTML e imagens. Elas podem apresentar vídeos, animações em várias tecnologias, como Macromedia™ Flash™ e a linguagem de modelagem de realidade virtual (VRML), áudio, Applets Java e componentes Microsoft™ ActiveX™ (MENDES, 2001; MENDES, 2002). Foram designados Objetos Web ("Web Objects") pelos autores estudados.

6.3 Características de Projetos Web

Os sistemas baseados em Web são compostos por Objetos Web, em que podemos classificar as páginas Web, as aplicações construídas em linguagem "script", como ASP, PHP, PERL, e ainda vídeos, animações e imagens com alta taxa de compressão (MENDES, 2002).

Nem todos os profissionais são especialistas em construir qualquer um desses objetos Web, por isso, em geral, as equipes são multidisciplinares. A grande demanda de mão-de-obra no auge dos investimentos em negócios na Internet emancipou muitos profissionais a executarem um trabalho, ainda pouco conhecido e sem uma organização e metodologia definidas. Posteriormente, surgiram muitas novas carreiras, em decorrência destes acontecimentos, tão rapidamente, que não foi possível equilibrar a evolução do mercado com as linhas de pesquisa.

A corrida pela presença na Internet foi marcada por uma democratização globalizada. Qualquer organização, ainda que desconhecida, poderia facilmente registrar um domínio[2] com o seu nome, hospedar uma aplicação de comércio eletrônico, um portal de informações ou outro tipo de aplicação que objetivasse o lucro.

Marcar presença antecipadamente em campanhas e outras ações publicitárias significou ganhar e sedimentar a preferência do público. Por isso, as aplicações Web se caracterizaram por terem o ciclo de desenvolvimento curto sem um processo definido. Reifer (2000) faz comparações entre o processo de desenvolvimento tradicional e o desenvolvimento de aplicações Web, conforme ilustra a tabela 6.1.

Comparação das características dos projetos Web e Tradicionais		
Característica	Desenvolvimento Tradicional	Desenvolvimento para Web
Objetivo	Construir produtos de software com qualidade com o menor custo	Disponibilizar no Mercado produtos de qualidade o mais rápido possível
Porte típico do Projeto	Médio a Grande (centenas de desenvolvedores)	Pequeno (3 a 5 profissionais)
Tempos de desenvolvimento típicos	10 a 18 meses	3 a 6 meses
Abordagens de Desenvolvimento utilizadas	Clássico, baseado em requisitos, em fases e/ou entregas incrementais, casos de uso, documentação.	Desenvolvimento rápido, prototipação, montagem de aplicações com componentes, RUP MBASE
Tecnologias utilizadas	Métodos orientados a objetos, geradores de aplicação, linguagem de programação modernas (C++), ferramentas CASE.	Métodos baseados em componentes, linguagens de 4ª e 5ª geração (HMTL, Java, e outros), visualização de movimentos e animações.
Processos empregados	Baseado no CMM	*Ad hoc*
Produtos desenvolvidos	Sistemas baseados em códigos-fonte, na maioria novos, alguma reutilização, muitas interfaces externas, freqüentemente aplicações complexas.	Sistemas baseados em objetos, muita reutilização, poucas interfaces externas, relativamente simples.
Perfil dos profissionais	Engenheiros de Software com mais de 5 anos de experiência em pelo menos dois domínios de aplicação.	"Designers" gráficos, engenheiros de software com 2 anos de experiência, e profissionais recém-formados.
Técnicas de Estimativa utilizadas	Analogia utilizando dados históricos de projetos, KSLOC ou baseado em pontos de função, EAP para projetos pequenos.	Analogia baseada na experiência, "projetar dentro dos recursos", EAP para projetos pequenos.

Tabela 6.1 - Comparação das características dos projetos Web e Tradicionais

[2] É um nome que serve para localizar e identificar conjuntos de computadores na Internet. O nome de domínio foi concebido com o objetivo de facilitar a memorização dos endereços de computadores na Internet. Sem ele, teríamos que memorizar uma sequência grande de números (FAPESP).

6.4 Contagem de Objetos Web

Os pontos de partida utilizados pelos pesquisadores foram os modelos de estimativas de esforço e os processos de desenvolvimento já utilizados em sistemas tradicionais e adaptados para as características dos sistemas baseados em Web (REIFER, 2000).

A contagem de Objetos Web tem sido objeto de estudo em várias linhas de pesquisa. Com o avanço dos recursos tecnológicos e das ferramentas para composição de aplicações e de edição de páginas, torna-se ainda mais desafiador definir processos de mensuração e níveis de complexidade para esses recursos em constante mudança.

MENDES et al. (2002) definiu dois tipos de aplicações Web:

- **Aplicações Web Hipermídia** – um tipo de aplicação bem diferente do convencional, que é caracterizada pela autoria e publicação de informações organizadas em nós (páginas web), cuja relação entre elas se dá através de ligações, denominadas "links", caracterizando uma estrutura de navegação. Para este caso, são utilizadas tecnologias como HTML, Javascript e arquivos multimídia (som, vídeo e animações). Os profissionais envolvidos nestes projetos não são somente técnicos, mas também, jornalistas, desenhistas, ilustradores e publicitários.
- **Aplicações de Software Web** – representa aplicações de software mais semelhantes aos tradicionais, mas que dependem de uma infra-estrutura típica da Web para serem executados. Nesta classificação se aplicam os sistemas de bancos de dados, comércio eletrônico ("e-commerce"), B2B e B2C, que são construídos com tecnologias, como DCOM, Microsoft™ ActiveX™, XML, PHP, DHTML, bancos de dados, J2EE, .NET, e componentes adquiridos comercialmente (COTS). Os profissionais envolvidos destacam-se, entre engenheiros de software, analistas de sistemas e gerentes de projeto.

A definição das métricas, que devem ser coletadas em um contexto de uma aplicação Web Hipermídia, foi estudada por Mendes et al., 2001, objetivando identificar quais são as variáveis influentes na estimativa de projetos Web e sua correlação com o esforço requerido. As técnicas utilizadas para essa avaliação foram: regressão linear e regressão por etapas (*stepwise*). As métricas foram classificadas por tamanho, nível de reuso e complexidade.

As métricas relacionadas ao tamanho e à complexidade estudadas por MENDES, 2001, foram:

- Aplicação
  - Número de páginas HTML ou SHTML
  - Número de arquivos de som ou vídeo
  - Número de Programas: CGI, Javascripts e Java Applets
  - Total de espaço em disco alocado por páginas HTML e SHTML (Mb)
  - Total de espaço em disco alocado por arquivos de som ou vídeo (Mb)
  - Número de linhas de código de todos os programas
  - Conectividade: número total de "links" estáticos internos
  - Densidade de Conectividade: quociente da Conectividade e número de páginas
  - Complexidade total da página:

$$\sum_{1}^{Totalpáginas} ComplexidadePágina / NumPaginas$$

- Página
  - Espaço em disco alocado por página HTML ou SHTML (Kb)
  - Complexidade de ligações da página: número de "links" por página
  - Complexidade da página: número de diferentes tipos de mídia presentes na página excluindo os textos
  - Complexidade gráfica: número de figuras utilizadas na página
  - Complexidade em áudio: número de arquivos de áudio utilizados na página
  - Complexidade em vídeo: número de arquivos de vídeos utilizados na página

    - Complexidade em animações: número de animações utilizadas na página
    - Complexidade de imagens digitalizadas: número de imagens digitalizadas e utilizadas na página
- Arquivos multimídia
    - Duração do áudio, vídeo ou animação (minutos)
    - Espaço em disco alocado pelo arquivo (Kbytes)
- Programa
    - Número de linhas de código
    - Número de linhas de comentários

As métricas relacionadas ao nível de reuso, foram:

- Aplicação
    - Número de arquivos multimídia reutilizados ou modificados
    - Número de programas reutilizados ou modificados
    - Total de espaço alocado em disco para todos os arquivos multimídias reutilizados ou modificados
    - Total de linhas de código reutilizados, em todos os programas reutilizados ou modificados.
- Programa
    - Número de linhas de código reutilizadas
    - Número de linhas de comentário reutilizadas

Mendes (2000) separou os esforços da mesma forma, considerando a construção da toda a aplicação, de cada página e de cada arquivo de áudio, vídeo ou animação.

Para a construção das páginas, foram considerados os tempos para composição do texto, estruturação da página, marcação de todos as ligações ("links") e alocação dos outros objetos Web. Para cada objeto Web, deve ser considerado o esforço de sua produção, eventualmente edição, e transformação para os formatos suportados. Os testes da integração dos objetos Web nas páginas e interligação entre elas devem ser

estimados, pois representam esforços significativos e que estão associados ao escopo global da aplicação.

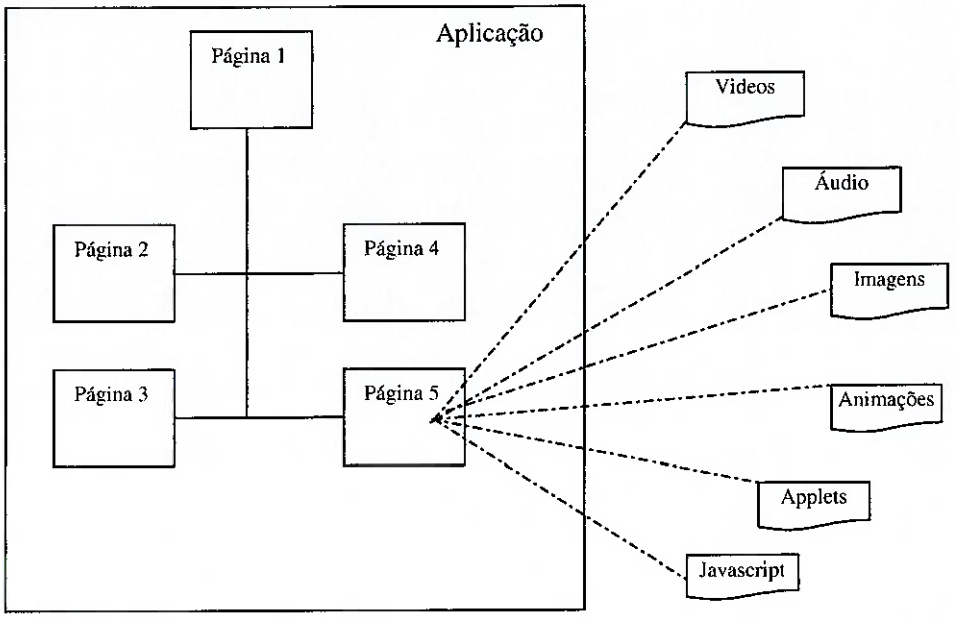

Figura 6.2 - Estrutura básica de uma aplicação Web hipermídia

As aplicações de Software Web, por sua vez, têm características que se aproximam das aplicações tradicionais. Em respeito ao número de linhas de código-fonte de programas, número de linhas de comentário, linguagens orientadas a objetos utilizadas.

6.5 Adaptação do Modelo

Os modelos de estimativas apresentados podem ser aplicados, com algumas adaptações, aos sistemas baseados em Web. A premissa maior passa a ser a calibração do modelo para refletir as características de um sistema Web. Como os sistemas Web são compostos de vários tipos de mídia, então, a contagem de pontos de função ou o número de linhas de código, por exemplo, pode ser feita apenas para essa parte da aplicação onde há programas não mesclados com HTML, ao contrário do que ocorre em linguagens script, como ASP e PHP. Mesmo nestes casos, se a

organização pode direcionar o desenvolvimento para a separação das funcionalidades de apresentação e de negócios, utilizando Padrões de Projeto ("Design Patterns"), que viabilizem a contagem de linhas de código.

CAPÍTULO 7

7 CONCLUSÕES

7.1 Considerações sobre o trabalho

Este trabalho descreveu os principais elementos que devem compor um programa de métricas. As diretrizes apontadas confirmaram que os modelos de estimativas de esforço, o histórico de projetos e os fatores de complexidade influenciam diretamente no cálculo das estimativas.

As diretrizes demonstraram que a estratégia de negócios da organização no programa de métricas é muito importante, visto que, sua implantação depende de altos investimentos. Além disso, a estratégia deve ser esclarecida para que os envolvidos estejam dispostos a contribuir com o programa, de tal forma que não tenham dúvidas quanto aos objetivos da organização de melhorar o processo de desenvolvimento. A definição dos dados a serem coletados deve ser conforme os objetivos da organização.

Os modelos apresentados não garantem uma margem de erro zero para as estimativas obtidas, e por isso, os autores sugerem a utilização de mais de um modelo ao mesmo tempo. A adoção de um modelo de estimativa mais complexo, como o COCOMO, exige um estudo aprofundado e profissionais habilitados, visto que, as atividades envolvidas com o processo de métricas demonstram mais de 10% de perda de produtividade, que deve ser recompensada com a melhoria nas estimativas realizadas.

O histórico de projetos mostrou-se um pré-requisito para os modelos de estimativas estudados.

A importância da coleta de dados é evidente, uma vez que a perda de dados na formação do histórico prejudica as estimativas de esforço e recursos. Esses dados devem ser validados para serem utilizados no apoio à decisão.

O modelo adotado deve sempre ser calibrado em função das tendências e dos objetivos da organização.

O processo de coleta de dados deve ser simples, assim como, os dados coletados, evitando que o histórico de projetos se torne incompleto.

A medida de complexidade de um sistema mostra a necessidade de utilizar ferramentas automatizadas. Sua influência no cálculo das estimativas é, assim como os fatores de escala, bem expressiva.

Os sistemas baseados em Web têm características diferentes dos sistemas chamados "tradicionais", e por isso, tem exigido pesquisas em torno dessas diferenças.

A engenharia de requisitos tem papel importante no controle de modificações de requisitos, auxiliando o gerenciamento durante o projeto, para que novas solicitações não prejudiquem as estimativas inicialmente obtidas. Estas relações podem ser mais bem exploradas em um trabalho futuro.

7.2 Trabalhos futuros

Foram evidenciados alguns assuntos que podem ser estudados a fim de dar continuidade a esse trabalho:

- Relacionar a complexidade de um sistema com os requisitos de software;
- Estudo da influência da engenharia de requisitos de software no processo de métricas;
- Métricas na utilização de "Design Patterns" em desenvolvimento de sistemas baseados em Web;
- Propor um modelo para contagem de Objetos Web.

8 ANEXO A – GLOSSÁRIO

Java Applet	Componente de software programado em Java voltado para execução embutido em programas navegadores de Internet (browsers).
Javascript	Linguagem script executada pelo programa navegador de Internet.
.NET	Plataforma da Microsoft para desenvolvimento de aplicações.

9 REFERÊNCIAS BIBLIOGRÁFICAS

ALBRECHT, A. J. **Measuring Application Development Productivity** – Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, Monterey, CA, Outubro, 1979.

ALBRECHT, A.J., GAFFNEY, J.E. – **Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation** – IEEE Transactions of Software Engineering, vol. SE-9, n° 6, Novembro, 1983.

AGUIAR, M – **Gerando Estimativas Confiáveis com COCOMO II e o Banco de Dados do ISBSG** – SUCESU-RJ – Dezembro 2002.

AMBLER, S.W., CONSTANTINE, L.L. **The Unified Process: Inception Phase** – CMP Books – 2000.

BIELAK, J. **Improving Size Estimates Using Historical Data** – IEEE Software – Novembro/Dezembro, 2000

BOEHM, B.W. **Software Engineering Economics** – 1981

BOEHM, B.W. et al. **Software Cost Estimation with COCOMO II** – Prentice Hall – 2000

BOEHM, B.W. **Anchoring the Software Process** – IEEE Software – vol.13, n° 4, Julho, 1996

BRAUDE, E.J. **Software Engineering: an Object-Oriented Perspective** – John Wiley & Sons – 2001.

FILGUEIRAS, L. – **Gerência de Projetos** - Material didático do curso de MBA Engenharia de Software – Escola Politécnica da Universidade de São Paulo – 2002. Não publicado.

HALE, J.; PARRISH, A.; DIXON, B.; SMITH, R. – **Enhancing the COCOMO Estimation Models** – IEEE Software – Novembro/Dezembro 2000.

HALSTEAD, M.H. – **Elements of Software Science** – Elsevier, North Holland, Dordrecht, The Netherlands, 1977

HIRAMA, K. – **Qualidade de Software** - Material didático do curso de MBA Engenharia de Software – Escola Politécnica da Universidade de São Paulo – 2002. Não publicado.

HUMPHREY, W.S. – **Managing the Software Process** – SEI Series in Software Engineering – Addison Wesley – 1989

JACOBSON, I., BOOCH, G., RUMBAUGH, J. – **The Unified Software Development Process** – Addison Wesley – 1999

KAN, S.H. - **Metrics and Models in Software Quality Engineering** – 2nd ed. – Pearson Education – 2003.

KULIK, Peter – **A Practical Approach to Software Metrics** – IT Pro Magazine – Jan/Feb 2000.

MAXWELL, K. D. **Collecting Data for Comparability: Benchmarking Software Development Productivity** – IEEE Software – Setembro/Outubro, 2001

MCCABE, T. J. – **A Complexity Measure** – IEEE Transactions on Software Engineering, vol. SE-2, n° 4, Dezembro, 1976

MELNIKOFF, S. – **Engenharia de Software** - Material didático do curso de MBA Engenharia de Software – Escola Politécnica da Universidade de São Paulo – 2002. Não publicado.

MENDES, E.; Mosley, N.; Counsell, S. – **Web Metrics – Estimating Design and Authoring Effort** – IEEE Multimedia, Web Engineering, Janeiro - Março, 2001.

MENDES, E., WATSON, I., TRIGGS, C., MOSLEY, N., COUNSELL, S. A **Comparative Study of Cost Estimation Models for Web Hypermedia Applications**, Empirical Software Engineering – an international journal, Kluwer Academic Publishers, 2002

PMI – **PMBOK 2000** – Tradução livre do PMBOK 2000 disponibilizada através da Internet pelo PMI-MG, Janeiro de 2002. http://pmimg.org.br, acesso em Fevereiro de 2002.

PRESSMAN, R.S. – **Software Engineering: A Practioner's Approach** – 5th ed. – McGraw Hill – 2001.

REIFER, D.J. **Web Development: Estimating Quick-to-Market Software** – IEEE Software, Novembro/Dezembro, 2000.

SILVEIRA, M. **Programa de Métricas: Medindo para poder melhorar** –BFPUG – Palestra ministrada em Agosto, 1999.

STRIKE, K., EMAM, K.E., MADHAVJI, N. **Software Cost Estimation with Incomplete Data** – IEEE Transactions on Software Engineering, vol.27, n° 10 - Outubro, 2001.

TAKATA, K. – **Redes de Computadores** - Material didático do curso de MBA Engenharia de Software – Escola Politécnica da Universidade de São Paulo – 2002. Não publicado.

UNIVERSIDADE DE SÃO PAULO. Escola Politécnica. Serviço de Bibliotecas. **Diretrizes para apresentação de dissertações e teses.** Serviço de Bibliotecas da EPUSP. 2ª ed. São Paulo, 2001.

VIJAKUMAR, R. – **Use of Historical Data in Software Cost Estimation** – Computing & Control Engineering Journal – Junho 1997.