Um modelo paramétrico de esforço para sistemas de informação baseados na arquitetura Web

Ana Carolina A. Carneiro *Eber A. Schmitz*

e-mail: carolina@posgrad.nce.ufrj.br, eber@nce.ufrj.br

NCE e IM / UFRJ
Caixa Postal 2324 – CEP 20001-970
Rio de Janeiro – RJ – Brasil

Resumo

Devido ao crescimento substancial do desenvolvimento de aplicações Web nos últimos tempos, tornou-se necessário o estudo de métodos, métricas e modelos de esforço para a Engenharia Web. Baseado neste princípio, esse trabalho aborda as características e particularidades da Web, propondo uma nova métrica de tamanho e um modelo para estimativa de esforço. O modelo foi validado com um conjunto de 11 projetos reais, onde mostrou uma grande correlação (R=0.99) entre o esforço real e os parâmetros do modelo.

Palavras chave: modelos de esforço, Engenharia Web, métricas Web, modelos Web

Abstract

Due to the substantial growth of the Web applications lately, it became necessary the study of methods, metrics and effort models for the Web Engineering. Based on this principle, this work approaches the Web characteristics and particularities, proposing a new size metric and a model for effort estimate. The model was validated with a group of 11 real projects, where it showed a good correlation (R=0.99) between the real effort and the parameters of the model.

Keywords**:** effort models, Web Engineering, Web metrics, Web models

1 Introdução

A Web (*World Wide Web*) é uma rede virtual, global, baseada em sistemas de informação hipertexto que utilizam a Internet como mecanismo de transporte para exibir páginas com informações gráficas, textuais, vídeos e até mesmo áudios [20].

A Web vem, cada dia mais, sendo utilizada pela população em geral e é um dos mais importantes desenvolvimentos da história computacional. A conseqüência desse contexto é um grande crescimento no desenvolvimento de aplicações Web.

O processo usado para a criação de aplicações Web de alta qualidade é chamado de Engenharia Web [15]. Embora a Engenharia Web aproveite conceitos e princípios da Engenharia de Software[15], suas aplicações apresentam características específicas, relacionadas na próxima Seção, que influenciam várias atividades da Engenharia Web, incluindo a atividade de estimativa de esforço de software.

A atividade de estimativa de esforço de projetos é uma atividade difícil e ainda muito deficiente nos projetos de software. Ela é a essência da dificuldade em controlar projetos de software [6].

Para um bom resultado na estimativa de esforço é muito importante tanto o uso de um modelo adequado como de métricas bem definidas e coletadas. Existem muitos modelos e métricas na literatura para estimativa de esforço e alguns deles estão apresentados na Seção 3.

Mas devido às características particulares de sistemas Web, torna-se importante a construção de um modelo de esforço e principalmente a definição de uma métrica de tamanho para aplicações Web.

O principal objetivo deste trabalho é o desenvolvimento de um modelo para o esforço de desenvolvimento de sistemas de informação Web. Como premissas essenciais deste modelo colocamos: (1) a simplicidade e (2) a possibilidade de ser utilizado em tempo de análise de requisitos. Para isso está sendo proposta uma nova métrica de tamanho, que engloba processamento da informação, páginas Web e complexidade da aplicação. O modelo apresenta também fatores de ajuste, baseados no produto, na equipe e no projeto. A estrutura do modelo, sua métrica de tamanho e fatores de ajuste estão descritos na Seção 4.

A métrica proposta foi aplicada em 11 projetos Web e as medidas obtidas foram usadas na validação do modelo de esforço. Os dados e resultados obtidos estão disponíveis na Seção 5. A Seção 6 mostra as conclusões deste trabalho.

2 A Engenharia Web e o processo de estimativa de esforço

Antes de apresentar um modelo de esforço para aplicações Web é necessário conhecer as particularidades dessas aplicações em relação às aplicações convencionais, principalmente quanto ao processo de estimativa de esforço.

Powel resume as principais diferenças entre as duas formas de engenharia de software quando afirma que "sistemas baseados em Web representam uma mistura entre publicidade e desenvolvimento de software, entre marketing e computação, entre relações externas e comunicações internas e entre arte e tecnologia" [14].

Para melhor entender essas diferenças segue uma lista de características particulares que as aplicações Web apresentam em relação às aplicações convencionais [15] : localizada em rede, dirigida a conteúdo, evolução contínua, estética, arquitetura três camadas – cliente (*browser*), servidor de aplicações (aplicação) e servidor de dados (banco de dados).

As características dos sistemas Web têm bastante influência no processo da Engenharia Web. Como as aplicações Web evoluem do conteúdo estático para dinâmico, com ambientes de aplicações dirigidos a usuário, cresce em importância a necessidade de aplicar um gerenciamento sólido e os princípios de Engenharia em projetos Web. Além disso a combinação de atividades técnicas e não técnicas representa um desafio para qualquer grupo de profissionais. Para evitar falhas, deve ocorrer um planejamento do projeto e os riscos devem ser considerados.

Parte desse planejamento está em estimar esforço, uma atividade de gerenciamento de projetos de software. Em teoria, a maioria das atividades de gerenciamento de projetos de software convencional se aplica a projetos Web. Mas na prática, o gerenciamento de projetos Web apresenta um maior número de desafios que impactam diretamente na estimativa de esforço: Primeiro porque o desenvolvimento de aplicações Web é uma área relativamente nova, contendo poucos dados históricos para serem usados na estimativa. E até hoje, poucas métricas de aplicações Web foram publicadas na literatura. [15]. Além disso, a estimativa de esforço de projetos deve ser baseada em projetos anteriores, mas quase toda aplicação Web tende a ser inovadora, oferecendo algo novo e diferente daqueles já existentes. Outra questão é que estimativa de esforço depende de um claro entendimento do escopo do projeto. E a evolução contínua, característica apresentada anteriormente, sugere que o escopo das aplicações Web seja um tanto vago.

Em virtude desses problemas e das particularidades de aplicações Web a realização de trabalhos como o aqui proposto é de grande importância no sentido de disponibilizar modelos e métricas direcionados à estimativa de esforço de sistemas baseados na arquitetura Web.

3 Métricas e modelos para estimativa de esforço

Estimativa de esforço de software é um processo em que se prevê a quantidade de esforço requerida para o desenvolvimento de um software. [7]

A estimativa de esforço de um projeto geralmente é feita a partir de uma das seguintes técnicas: analogia, estimativa paramétrica, estimativa *bottom-up* ou ferramentas computadorizadas (simulação, estatística,..). As estimativas paramétricas utilizam um modelo de esforço [13].

Em geral, os modelos de esforço dependem de um fator de esforço principal que é o "tamanho" do software e de um conjunto de fatores de ajuste, relacionados tanto ao produto como ao projeto [7]. A partir do tamanho do software, é possível estimar esforço e consequentemente custo e prazo.

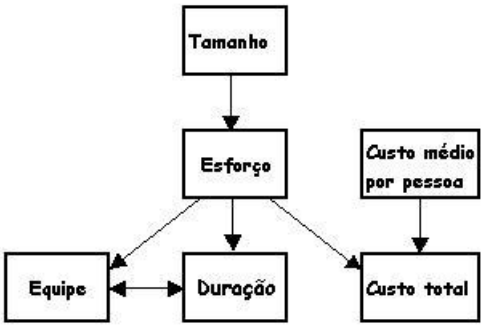

Figura 3.1: Relação entre tamanho do sistema e modelos de esforço, duração, equipe e custo [12]

Os fatores de ajuste expressam características de complexidade do produto, de projeto, do processo de desenvolvimento e dos recursos. Eles são usados para ajustar a estimativa preliminar fornecida pelo fator de esforço principal.

Normalmente os modelos de esforço são derivados da análise de dados de projetos passados. A estrutura de um típico modelo de esforço (sem fatores de ajuste) tem a forma: [10]

$E = A + B * T^C$, onde

A, B e C são constantes derivadas empiricamente,

E é o esforço e T é o tamanho estimado do software (LOC ou PF por exemplo).

3.1 Métricas de tamanho

Como o tamanho é o principal fator de esforço na maioria dos modelos, é importante utilizar a métrica de tamanho mais adequada ao produto a ser estimado. As principais maneiras de se medir tamanho de software são linhas de código e pontos de função: [7]

3.1.1 Linhas de Código

A métrica mais comum baseada no código fonte é o número de linhas de código (*Lines of Code* - LOC). Embora essa medida já tenha sido muito usada, dois tipos de restrições a impedem de ser utilizada plenamente.

A primeira tem a ver com a mudança das linguagens de programação, que utilizam recursos visuais, gerando código automaticamente, e técnicas de orientação a objetos, reutilizando código.

A segunda restrição diz respeito à fase do projeto onde a medida deve ser estimada. Linhas de código é uma medida que só é conhecida após a implementação do software, tendo-se pouca idéia de seu valor em tempo de análise de requisitos.

3.1.2 Pontos de Função

Essa métrica é baseada na funcionalidade do sistema e é a mais usada hoje em dia. A métrica de pontos de função pode ser aplicada nas fases iniciais de desenvolvimento, e portanto permite estimar o tamanho do sistema antes dele estar pronto. Existem adaptações do método original proposto por Albrecht [2], como o método Mark II de Symons [18].

Apesar de ser uma métrica de tamanho bem sucedida e muito utilizada nos projetos de software, ela possui algumas restrições que justificam a necessidade da definição de novas métricas para aplicações Web:

- É baseada no tamanho do processamento da informação. Em sistemas de informação Web, é importante considerar também o tamanho da interface (páginas), já que este é um fator que influencia muito o esforço de desenvolvimento de aplicações Web. Esta influência se dá devido à necessidade de transformar o conteúdo estático das páginas em dinâmico.
- Os pesos utilizados pela métrica foram encontrados há mais de 20 anos atrás a partir de projetos cujas características diferem muito das características de sistemas Web.
- Não é uma métrica simples, demandando um tempo considerável para ser coletada. Como o tempo de desenvolvimento de aplicações Web precisa , em geral, ser o mais rápido possível, a demanda desse tempo não se torna adequada.

3.2 Modelos de esforço

Através dessas duas principais métricas de tamanho, foram propostos diversos modelos de esforço, entre eles o COCOMO II e o SLIM:

O modelo COCOMO (Constructive Cost Model) foi originalmente publicado em [4] e se tornou um dos mais populares modelos paramétricos de esforço da década de 80. Com o passar do tempo ele não se mostrou mais adequado aos novos processos de desenvolvimento de software, surgindo a necessidade da criação do modelo COCOMOII [3].

O COCOMO II possui três sub-modelos, *Application Composition, Early Design e Post Architeture*. O modelo *Application Composition* é usado para estimar esforço e prazo de projetos de rápido desenvolvimento. Ele usa como métrica de tamanho *Object Points* ao invés da tradicional Linhas de Código (LOC). *Early Design* e *Post Architeture* são modelos usados nos estágios iniciais e durante o desenvolvimento do software, respectivamente.

Putnam desenvolveu um modelo chamado SLIM [16] para ser aplicado a projetos com mais de 70.000 linhas de código. O modelo SLIM é expresso em duas equações descrevendo a relação entre o esforço de desenvolvimento e o prazo. A primeira equação é chamada de *software equation* e a segunda de *manpower-buildup*.

3.3 Modelos de esforço para aplicações Web

Até hoje foram definidas poucas métricas e modelos de esforço para aplicações Web [15]. Nessa Seção serão apresentados dois modelos que estimam o esforço para o desenvolvimento de aplicações Web:

Reifer propôs uma métrica de tamanho chamada Web Objects e uma adaptação do modelo COCOMO II chamada WebMo para estimar esforço e duração de aplicações Web. [17]. Para medir o tamanho do sistema de informação Reifer usa como métrica o número de Object Points [3].

O tamanho da aplicação é calculado com a soma do número de Web Objects e o ajuste desse valor com o peso de complexidade. A descrição dos Web Objects e seus respectivos pesos podem ser encontrados na referência. O esforço é estimado de acordo com a equação:

$E = A \P Cdi\ S^{P1}$, onde

E é o esforço estimado

A e P1 são constantes definidas empiricamente

Cdi são multiplicadores de esforço, baseados nos multiplicadores do COCOMO II

S é o número de Web Objects

Mendes, Mosley e Counsell também proporam um modelo de esforço para aplicações Web [11]. Como no modelo de Reifer, foram sugeridas algumas métricas de tamanho nesse modelo. As métricas coletadas medem atributos categorizados em diferentes tipos de entidades: aplicação (número de páginas, número de mídias, número de programas,...), página (complexidade dos links, complexidade das imagens, complexidade dos vídeos,...), mídia (duração e alocação) e programa (número de linhas de código, número de linhas de comentário).

O modelo foi definido com a regressão (linear e múltipla) de variáveis correspondentes às métricas em questão. Foram gerados diferentes modelos para cada categoria usada: aplicação, página, mídia e programa. O modelo obtido para a aplicação caracteriza uma abordagem *top-down*, enquanto os modelos obtidos para página, mídia e programa representam a abordagem *bottom-up*. Os autores não disponibilizaram os pesos de cada variável do modelo na referência.

Os dois modelos apresentam como principais métricas componentes Web (scripts, vídeos, imagens, áudios, htmls) dando uma menor importância ao processamento da informação e às funcionalidades do sistema. O modelo proposto neste trabalho mostra uma abordagem diferente e está apresentado na próxima Seção.

4 Estrutura e descrição do modelo proposto

Um sistema de informação Web, como já foi dito, apresenta no seu desenvolvimento atividades técnicas (processamento de informações, implementações de regras de negócio) e não técnicas (criação de páginas estáticas, vídeos, imagens). Como em grande parte dos projetos as atividades não técnicas são executadas por outros profissionais e estão relacionadas muito mais à criatividade do que à Engenharia de Software, o método de tamanho e o modelo de esforço deste trabalho não incluirão o esforço dessas atividades.

As métricas aqui propostas focam na funcionalidade e complexidade do sistema de informação. Como já existem métricas e modelos de esforço para produção de hipertextos na literatura [17] [11], é possível estimar o esforço dos dois tipos de atividades separadamente:

Et = Epag + **Efunc**, onde

Et é o esforço total para construção do sistema Web

Epag é o esforço para construção das páginas ainda estáticas e seus componentes

Efunc é o esforço para implementar as funcionalidades do sistema nas páginas Web, estimado pelo modelo aqui proposto.

Como já foi dito na Seção anterior, a estrutura típica de um modelo de esforço é $E = A + B\ X\ T^C$. Nesse trabalho foi proposta uma estrutura de modelo mais compreensível e intuitiva, que se aplica à construção de muitos produtos do nosso dia a dia.

A idéia de construir um modelo simples veio da necessidade de uma estimativa rápida, condizente com o tempo de desenvolvimento de um sistema Web. A estrutura de modelo proposta é baseada na seguinte premissa:

$$\text{Esforço} = \frac{\text{Tamanho}}{\text{Produtividade}}$$

Chamamos de produtividade o número de pontos que é desenvolvido numa organização por uma pessoa em um mês. Dessa maneira o modelo aqui definido pode ser usado por várias organizações, desde que essa produtividade seja recalculada.

Mas no processo de construção de um software a produtividade de uma organização não é constante em todos os projetos. Ela varia de acordo com fatores da equipe que participa das atividades, do processo usado para construção do software e do ambiente do projeto. Além disso o tamanho de um software deve ser ajustado de acordo com os fatores técnicos do produto. Dessa maneira a estrutura do modelo de esforço proposta é

$$E = \frac{T * \P Fpr}{P * \P Feq * \P Fpj}, \text{ onde}$$

E é o esforço em pessoas-mês

T é o tamanho em PFW (pontos de função Web)

Fpr são os fatores técnicos e de complexidade do produto que influenciam no seu tamanho.

P é a constante de produtividade definida empiricamente em PFW / pessoas-mês

Feq e Fpj representam os fatores que influenciam a produtividade – que são os fatores da equipe que participa do projeto e das características próprias do projeto de desenvolvimento.

O processo para estimativa do esforço está representado no diagrama de atividades da Figura 4.1. A próxima Seção mostra um detalhamento de como encontrar os valores das variáveis desse modelo.

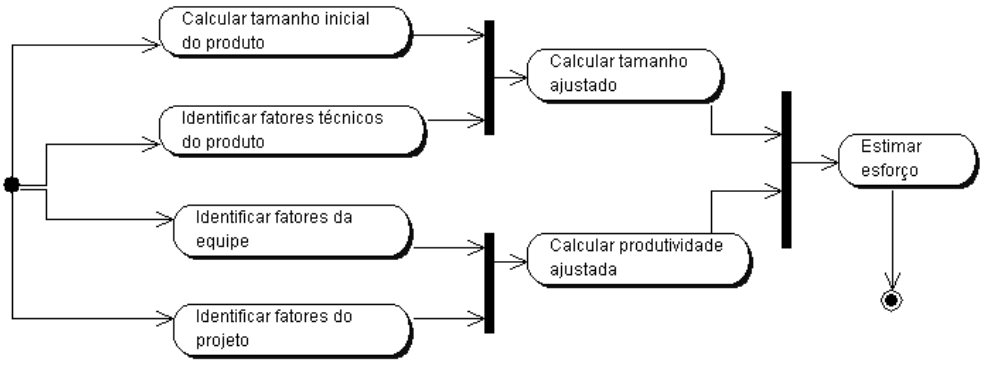

Figura 4.1: Processo de estimativa de esforço

4.1 Tamanho (T)

As métricas e o processo de medição de tamanho de um sistema de informação Web foram baseados nas camadas da arquitetura Web, já apresentadas na segunda Seção deste trabalho .

A camada cliente representa as operações de interface com o usuário para entrada e apresentação dos dados através das páginas Web, determinando o esforço de transformar as páginas estáticas em dinâmicas.

A camada de aplicação representa o grau de complexidade das operações de controle e de regras de negócio da aplicação, determinando o esforço de implementá-las.

A camada de dados representa o processamento dos dados da aplicação, determinando o esforço de recuperar e armazenar dados na base de dados.

A partir dessa abordagem foi proposto que o fator principal do tamanho de um sistema de informação Web seja o seu número de páginas, classificadas de acordo com : (1) o processamento de informação que ela provê (2) a complexidade desse processamento.

O processo de medição proposto é baseado em casos de uso. Vale a pena ressaltar que a aplicação não precisa ser desenvolvida segundo o paradigma orientado a objetos para usar esse método, já que a definição de casos de uso de uma aplicação é independente do estilo de programação utilizado.

A definição de requisitos funcionais usando a técnica de casos de uso vem sendo muito utilizada nos projetos e facilitará a coleta das métricas de tamanho. Segundo Carol Dekkers, na medida que imaginamos um cenário, temos maior facilidade de estimar as métricas:

"*Casos de uso apresentam requisitos lógicos do usuário em um formato completo e facilmente digerido que torna a contagem de pontos de função simples*" [5]

Utilizaremos, dessa forma uma abordagem *bottom-up*, onde o tamanho de cada caso de uso é utilizado para estimar o tamanho total do sistema.

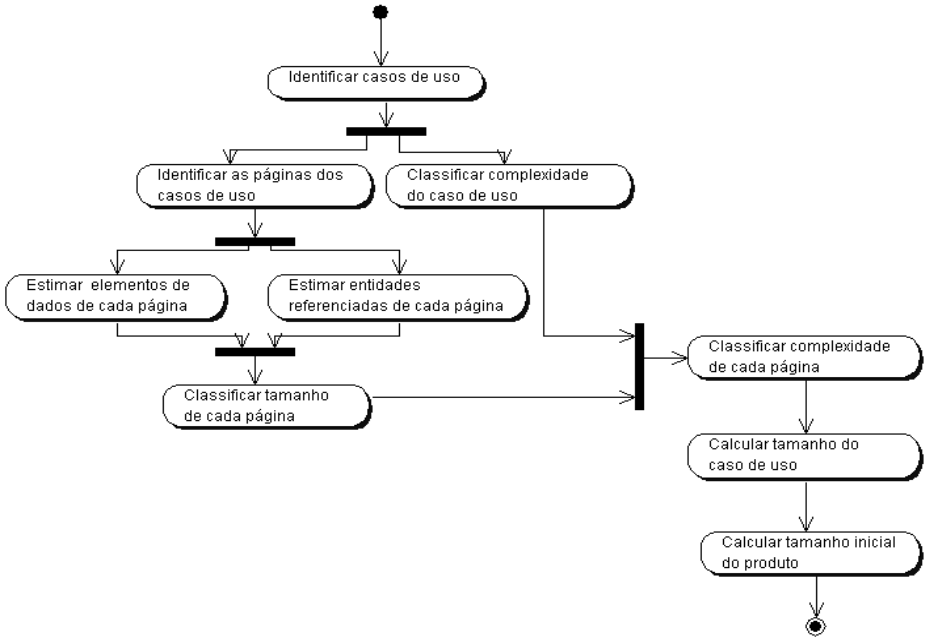

Figura 4.2: Processo de medição de tamanho

O processo para medição de tamanho de cada caso de uso e consequentemente de todo o sistema está apresentado na Figura 4.2 e descrito a seguir.

a. Identificar os casos de uso da aplicação

Como a estimativa de esforço é feita ainda na fase de análise de requisitos da aplicação, os casos de uso identificados não precisam ser detalhados e/ou fatorados. O importante é identificar os casos de uso principais e as possíveis reutilizações.

b. Classificar a complexidade dos casos de uso em simples, média e alta

Depois de identificados, os casos de uso devem ser classificados quanto à sua complexidade. A complexidade não deve ser vista como um fator de ajuste e sim como um componente do tamanho de uma aplicação. Cada caso de uso deve ser classificado em simples, médio e complexo. A complexidade de ser avaliada de acordo com três tipos de operações: gerenciamento de dados, interface e controle. A definição da complexidade de um caso de uso a partir da avaliação dessas operações deve ser feita de acordo com o método a seguir.

b.1 Classificar em simples, médio e complexo cada item da Tabela 4.1, de acordo com as descrições nela apresentadas. Parte desta tabela (operações de controle e gerenciamento de dados) foi construída com base no fator de ajuste CPLX do modelo COCOMO II [3].

b.2 Classificar o caso de uso em simples, médio ou complexo seguindo o seguinte critério :

Simples: 70% ou mais dos itens são simples. Nenhum item complexo

Complexo : 30% ou mais dos itens são complexos.

Médio: Caso contrário

Gustav Karner também propôs um método para classificação de casos de uso em simples, médio e complexo baseado no número de transações [8]. Esta métrica categoriza na verdade o tamanho do caso de uso e não a sua complexidade. Por outro lado, na métrica proposta, o tamanho de um caso de uso é calculado a partir da complexidade das páginas nele contidas. A complexidade da página é obtida combinando-se o tamanho da página à complexidade do caso de uso à ela associado (passo g), permitindo a definição de uma métrica mais apropriada a casos de uso Web .

c. Identificar as páginas de cada caso de uso

Para cada caso de uso identificado, devem ser identificadas as páginas nele envolvidas. Se uma página aparece em mais de um caso de uso ela deve ser contabilizada apenas uma vez e associada ao caso de uso mais complexo. Um e-mail gerado e enviado pela aplicação, também pode ser considerado uma página.

Operações de controle			
	Simples	Médio	Complexo
Aninhamento	Passos diretos, muito pouco aninhados	Maioria de aninhamentos simples	Estrutura altamente aninhada
Passos	Maioria Simples	Médio, com alguns complexos	Muitos passos complexos
Código recursivo	Não		Sim
Sincronização de tarefas	Não		Sim
Controle em tempo real	Não		Sim
Controle de pilha ou fila	Não		Sim
Tabela de decisão	Não	Sim, simples	Sim, complexas
Operações de gerenciamento de dados			
	Simples	Médio	Complexo
Consultas	Simples a moderadas	Um pouco complexas	Complexas, otimizadas
Atualizações	Simples a moderadas	Um pouco complexas	Complexas
Armazenamento em memória	Simples	Médio	Complexo
Triggers	Não	Simples	Complexos
Mudanças estruturais	Não	Simples	Complexas
Integração com outros sistemas	Não	Simples	Complexa
Aplicação distribuída	Não		Sim
Base de dados distribuída	Não		Sim
Operações de gerenciamento de Interface			
	Simples	Médio	Complexo
Atualização dinâmica em formulários	Não	Moderadas, de acordo com operações do usuário	Muitas , de acordo com operações do usuário
Validação de campos em formulários	Não	Alguns campos	Muitos campos
Apresentação da página X Dados de saída	Independentes	Parte da apresentação dependente dos dados	Maioria da apresentação dependente dos dados
Formatação de dados para apresentação	Não	Alguns dados, formatação simples	Maioria dos dados, formatação complexa

Tabela 4.1: Classificação de complexidade dos casos de uso

d. Estimar número de elementos de dados de cada página

Para estimar o número de elementos de dados de uma página deve-se somar o número de elementos de entrada ao número de elementos de saída para cada funcionalidade da página (inserção, edição, remoção, consulta,...).

Um elemento de dado de entrada é um elemento de dado informado para a aplicação através da página para ser armazenado ou para servir de critério para algum processamento.

Um elemento de dado de saída é um elemento de dado recuperado ou gerado pela aplicação que será apresentado na página. Se um mesmo elemento de dado aparece como entrada e como saída na página, ele deve ser contado duas vezes, como entrada e como saída.

e. Estimar número de entidades referenciadas de cada página

Para estimar o número de entidades referenciadas em uma página deve-se contar o número de entidades referenciadas nas operações de entrada e saída para cada funcionalidade da página. O conceito de entidade corresponde ao conceito de entidade do modelo de dados de domínio do sistema.

f. Classificar tamanho da página em pequeno, médio ou grande

O tamanho das páginas (pequeno, médio ou grande) será definido de acordo com o número de elementos de dados e o número de entidades referenciadas, já medidos anteriormente.

Uma vez contabilizadas essas medidas, deve-se classificar cada página em pequena, média e grande de acordo com a Tabela 4.2, a seguir.

	1-5 elementos	6-10 elementos	> 10 elementos
0-2 entidades	PEQUENA	PEQUENA	MÉDIA
3-4 entidades	PEQUENA	MÉDIA	GRANDE
> 4 entidades	MÉDIA	GRANDE	GRANDE

Tabela 4.2: Classificação de tamanho das páginas

g. Classificar complexidade de cada página

Para obtermos a classificação final de complexidade das páginas de um caso de uso temos que associar a complexidade do caso de uso, já definida a partir do passo b, ao tamanho das suas páginas, estimado no passo anterior. Essa associação deve ser feita de acordo com a Tabela 4.3.

h. Calcular tamanho de cada caso de uso

Obtida a classificação final de todas as páginas de um caso de uso, seu tamanho não ajustado (Tc) é estimado pela equação:

Tc= w1*Ps + w2*Pm + w3*Pc, onde

Tc é o tamanho do caso de uso

Ps, Pm e Pc é o número de páginas simples, médias e complexas, respectivamente

w1, w2, w3 é o peso de cada classificação de página, respectivamente

i. Calcular o tamanho inicial do produto

Para se obter o tamanho não ajustado de um sistema de informação Web basta somar os tamanhos de cada um dos casos de uso identificados.

	Tamanho da Página			
Complexidade do Caso de Uso		Pequena	Média	Grande
	Simples	SIMPLES	SIMPLES	MÉDIA
	Médio	SIMPLES	MÉDIA	COMPLEXA
	Complexo	MÉDIA	COMPLEXA	COMPLEXA

Tabela 4.3: Classificação de complexidade das páginas

4.2 Fatores técnicos (Fpr)

Uma vez calculado o tamanho não ajustado do sistema, é necessário avaliar alguns fatores técnicos para ajustar esse tamanho preliminar. No nosso método a maioria desses fatores representa um subconjunto dos multiplicadores de esforço do modelo COCOMO II [3]. Neste subconjunto encontram-se os multiplicadores de esforço que também se aplicam a sistemas de informação Web.

Foram adicionados mais dois fatores de ajuste no modelo: PERF, que avalia o desempenho exigido pelo produto e SECU, que avalia a segurança exigida. Esses fatores são muito importantes para sistemas de informação Web. PERF porque na medida que na arquitetura Web o servidor de aplicações é remoto, muitas vezes o desempenho da aplicação não corresponde aos requisitos do usuário, gerando uma necessidade de procedimentos para melhoria de desempenho. E SECU porque uma vez que a aplicação está numa rede, procedimentos mais rigorosos para garantia de segurança devem ser estabelecidos.

Resumindo, os fatores para ajuste do tamanho de um sistema de informação Web são os seguintes:

RELY – efeito da falha do software. Se a falha gerar apenas inconveniência terá um RELY baixo, já quando a falha implica em risco a vida humana seu RELY é alto.

DATA – tamanho (em relação à aplicação) e complexidade do modelo de dados

DOCU – nível de documentação requerida

RUSE – produto ou parte do produto desenvolvido com intuito de ser reutilizado

PERF – grau da desempenho exigida pelo produto. Quando um bom desempenho é fundamental e um fator crítico para a aplicação seu valor de PERF será alto, caso seja apenas conveniente seu PERF será baixo.

SECU – grau de segurança exigida pelo produto. Se a segurança for apenas um fator de conveniência, baixo SECU, caso a segurança seja fundamental e um fator crítico do sistema, SECU será alto

Esses fatores devem ser classificados numa escala ordinal que vai desde “muito baixo” a “muito alto”, de acordo com a Tabela 4.4. A partir dessa classificação, um fator multiplicativo deve ser atribuído ao tamanho preliminar. A Tabela 4.4 mostra a regra de classificação dos fatores e seus respectivos multiplicadores

4.3 Fatores de produtividade (¶Feq e ¶Fpj)

A produtividade média de uma organização (P) deve ser calculada empiricamente a partir de seus dados históricos. Mas essa produtividade pode não ser a mesma em todos os projetos devido a fatores da equipe e do próprio projeto.

Esses fatores foram selecionados a partir dos multiplicadores de esforço e dos fatores de escala do modelo COCOMO II [3], com a inclusão de um fator chamado EVOL, que avalia o risco de volatilidade do escopo. Como uma das características da Engenharia Web é a evolução contínua, o que sugere que o escopo das aplicações seja um tanto vago, se torna importantíssimo avaliar esse fator.

Os fatores de produtividade estão relacionados a fatores da equipe e do projeto e também devem ser classificados numa escala de “muito baixo” a “muito alto”. A partir dessa classificação, um fator multiplicativo deve ser atribuído à produtividade média definida. A Tabela 4.5 orienta essa classificação.

4.3.1 Fatores da equipe

PERS – capacidade (habilidade, conhecimento) da equipe

PREX – experiência da equipe no desenvolvimento de aplicações, na plataforma, na linguagem e nas ferramentas

TEAM - Habilidade, coesão do trabalho em equipe e continuidade da equipe

4.3.2 Fatores do projeto

FCIL – facilidades das ferramentas e equipamentos utilizados

PDIF – volatilidade e dificuldade da plataforma (desenvolvimento e produção)

PEFF – maturidade e eficiência do processo de desenvolvimento

EVOL - volatilidade do escopo

	Muito baixo	**Baixo**	**Normal**	**Alto**	**Muito Alto**
RELY	Efeito apenas inconveniente Fator: 0,82	Baixo, fácil de recuperar Fator: 0,92	Moderado Fator: 1	Alta perda financeira Fator: 1,1	Risco vida humana Fator:1,26
DATA		Pequeno, simples Fator:0,9	Moderado Fator:1	Grande, complexo Fator:1,14	Muito Grande, complexo Fator:1,28
DOCU	Muitos ciclos de vida não necessitam Fator:0,81	Alguns ciclos de vida não necessitam Fator:0,91	Os ciclos de vida necessitam dentro da normalidade Fator:1	Excessiva para as necessidades dos ciclos de vida Fator:1,11	Muito excessiva para a necessidade dos ciclos de vida Fator:1,23
RUSE		Nenhum Fator:0,95	Implementado para uma parte da aplicação Fator:1	Implementado para a aplicação inteira Fator:1,07	Implementado para a linha do produto Fator:1,15
PERF	Nenhuma Fator: 0,82	Pouca (não é tão importante). Nenhuma ação nova será necessária para cumprir os requisitos Fator:0,92	Moderada (é importante) Algum procedimento novo, mas simples, será necessário para cumprir os requisitos Fator: 1	Alta (é muito importante) Serão necessárias ações, que exigirão considerável esforço, para cumprir os requisitos de performance Fator:1,1	Muito Alta (é essencial) Requisitos de performance estabelecidos são rigorosos. Necessidade de tarefas de análise de performance e/ou ferramentas Fator:1,26
SECU	Nenhuma Fator:0,82	Pouca (não é tão importante) Nenhuma ação nova será necessária para cumprir os requisitos Fator:0,92	Moderada (é importante) Algum procedimento novo, mas simples, será necessário para cumprir os requisitos Fator:1	Alta (é muito importante) Serão necessárias ações, que exigirão considerável esforço, para cumprir os requisitos de segurança Fator:1,1	Muito Alta (é essencial) Requisitos de segurança estabelecidos rigorosos. Necessidade de tarefas de grande esforço para cumprir os requisitos de segurança Fator:1,26

Tabela 4.4 : Classificação dos fatores técnicos do produto

	Muito baixo	Baixo	Normal	Alto	Muito Alto
PERS	Capacidade media de 39% Fator: 0,74	Capacidade media de 45% Fator:0,87	Capacidade media de 55% Fator:1	Capacidade media de 65% Fator:1,15	Capacidade media de 75% Fator:1,31
PREX	Experiência de 3 meses a 5 meses Fator:0,83	Experiência em torno de 9 meses Fator:0,91	Experiência em torno de um ano Fator:1	Experiência de 2 a 4 anos Fator:1,11	Experiência maior que 4 anos Fator:1,21
TEAM	Interações na equipe muito difíceis 30% de volatilidade da equipe Fator:0,78	Interações na equipe um pouco difíceis 20% de volatilidade da equipe Fator:0,89	Interações na equipe basicamente cooperativas 12% de volatilidade da equipe Fator:1	Interações amplamente cooperativas 9% de volatilidade da equipe Fator:1,11	Interações muito cooperativas 5% de volatilidade da equipe Fator:1,23
FCIL	Algum suporte de ferramenta Fator:0,85	Ferramenta case simples Fator:0,91	Ferramentas para o ciclo de vida básico Fator:1	Bom. Ferramentas moderadamente integradas Fator:1,11	Forte. Ferramentas bem integradas Fator:1,28
PDIF		Poucas mudanças na plataforma. Serviço de rede rápido. Poucas limitações computacionais Grande experiência no servidor aplicações usado Fator:0,85	Plataforma estável Performance da rede aceitável Experiência no servidor aplicações usado. Fator:1	Mudanças freqüentes na plataforma Rede com pouca performance Problemas causados por falta de recursos computacionais Pouca experiência no servidor aplicações. Fator:0,9	Plataforma muito volátil. Rede com pouca performance Muitos problemas causados por falta de recursos computacionais Nenhuma ou muito pouca experiência no servidor de aplicações. Fator:0,8
PEFF	Totalmente ad hoc, confuso. O projeto depende das pessoas Fator:0,78	Processo baseado nos projetos. O projeto depende do gerenciamento Fator:0,85	Processo adaptado para o projeto. O processo serve de guia Fator:1	Processo eficiente, indo ao encontro do trabalho. Fator:1,1	Processo efetivo, vai ao encontro dos objetivos. Todos usam e acreditam no processo. Fator:1,22
EVOL	Escopo sem mudanças, muito estável e bem definido Fator:1,22	Escopo com raras mudanças, bem definido Fator:1,1	Escopo com mudanças aceitáveis, causando retrabalho dentro do esperado Fator:1	Escopo com muitas mudanças, gerando retrabalhos além do esperado Fator:0,85	Escopo não estável, com grandes e constantes mudanças, gerando retrabalhos não aceitáveis Fator:0,78

Tabela 4.5 : Classificação dos fatores da equipe e do projeto

5 Calibração e validação do modelo

Uma vez estabelecida a regra de contagem e as métricas para estimativa de esforço de sistemas de informação Web, o próximo passo foi a validação do modelo e a definição das variáveis empíricas (w1,w2,w3 e P). Para isso foram usados como piloto dados históricos de onze projetos Web de uma empresa de consultoria em informática. As medidas coletadas estão sumarizadas na Tabela 5.1.

As medidas de tamanho foram obtidas a partir da análise dos próprios produtos, já que muitos deles estavam com documentação deficiente. Além disso, foi necessário realizar reuniões com os participantes de cada projeto para obter um conhecimento maior sobre os produtos e coletar os fatores de ajuste. Como a empresa dispunha de um sistema de cadastro de atividades, por projeto e consultor, o número de horas gastas em cada projeto foi facilmente obtido.

Essas medidas foram exportadas para uma ferramenta de análise estatística chamada Statistica [19]. O método utilizado pela ferramenta para encontrar as variáveis empíricas foi o método de estimativa Quasi-Newton [9]. Após encontradas, as variáveis foram normalizadas por w1. O valor do coeficiente de correlação(R) obtido para o modelo encontrado foi 0,9983. Os valores encontrados para as variáveis empíricas estão apresentados na Tabela 5.2.

Apesar de onze projetos não ser um número grande para termos um alto grau de confiança no modelo, já podemos ter um indicador de que as métricas e consequentemente o modelo foram bem estabelecidos. O gráfico da Figura 5.1 representa a relação esforço previsto pelo modelo X esforço realizado e o da Figura 5.2, a relação tamanho sem ajuste X esforço real.

Projeto	Ps	Pm	Pc	¶Fpr	¶Feq*□Fpj	Esforço(hm)*
1	10	2	10	1,53	1,25	6,82
2	40	1	0	0,89	1,10	3,12
3	23	7	5	0,78	0,72	5,62
4	150	26	19	1,05	0,57	49,35
5	13	1	0	0,96	0,89	1,77
6	45	10	3	0,96	0,48	13,34
7	76	21	5	0,95	1,28	11,58
8	0	4	2	0,9	0,69	1,82
9	9	6	10	1,19	1,25	5,64
10	45	6	9	0,86	0,71	9,82
11	6	0	0	0,79	1,05	0,47

Tabela 5.1: medidas coletadas em projetos Web

Os valores encontrados para as variáveis empíricas (principalmente P) devem ser recalculados com o histórico da própria organização, visto que a produtividade sofre alterações com o tempo e não é a mesma em todas as organizações. Para a organização piloto os valores encontrados foram:

Variável	Valor
P	11,28
W1	1
w2	2,58
w3	4,38

Tabela 5.2: valores das variáveis empíricas

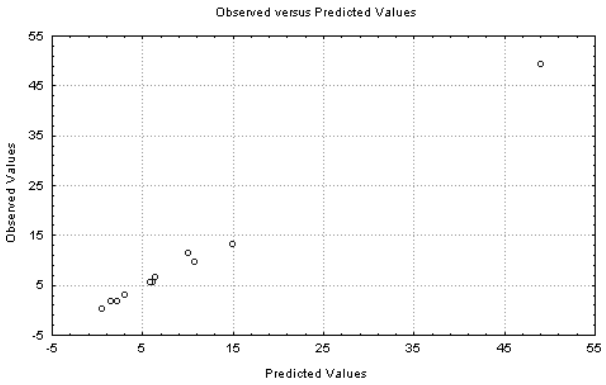

Figura 5.1: Esforço previsto X esforço real

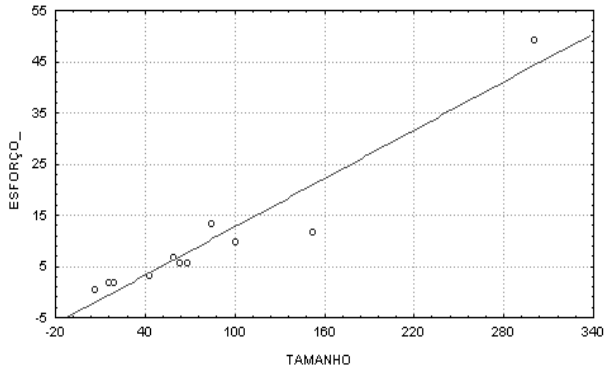

Figura 5.2:Tamanho não ajustado X esforço real

6 Conclusões

A definição de um modelo de esforço é um trabalho difícil e que requer muito tempo para ser feito. Isso se deve ao fato de precisarmos definir métricas adequadas, possuir dados confiáveis e coletar as medidas corretamente, para então chegarmos a um modelo matemático em função dessas métricas.

Apesar dos testes iniciais terem sido realizados com poucos projetos, o resultado obtido através dos dados foi um bom indicador de que o modelo foi bem definido e de que as métricas foram bem estabelecidas.

A idéia de se ter um modelo simples e intuitivo facilita a definição empírica das variáveis para cada organização que o for utilizar. A produtividade (P) de cada organização deve ser sempre calculada à medida que o histórico dos projetos muda, fazendo com que a estimativa vá ao encontro da realidade atual da organização. Ou seja, uma eficiente estimativa de esforço não depende unicamente do modelo aqui apresentado, mas também de um histórico de projetos confiável.

Como parte dos trabalhos futuros está a validação desse modelo em um conjunto maior de projetos e o desenvolvimento de uma ferramenta para coleta e armazenamento das métricas aqui apresentadas e para estimativa das variáveis empíricas do modelo e consequentemente do esforço do projeto.

Referências

[1] Albrecht, Allan J.; Gaffney, John E.; “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation”, IEEE Transactions on Software Engineering, Vol 9, 1983

[2] Albrecht, A. J.; “Measuring application development productivity”, Guide/Share Application Develop. Symp. Proc.,Outubro, 1979

[3] Boehm, B. et al.; “Software Cost Estimation with Cocomo II, Prentice-Hall, Upper Saddle River, N.J., 2000.

[4] Boehm, B.; “Software Engineering Economics”, Prentice Hall, 1981

[5] Dekkers, Carol A. ;”Use Cases and Function Points – Where’s the fit?”, IT Metrics Strategies, Janeiro, 1999.

[6] DeMarco, Tom; “Controle de Projetos de Software”, Editora Campus,1989

[7] Johnson, K. “Software Cost Estimation : Metrics and Models”, University of Calgary, 1998

[8]Karner, Gustav; “Use Case Points - Resource Estimation for Objectory Projects”, Objective Systems SF AB (copyright owned by Rational Software), 1993

[9] Martinez, José M. “Algorithms for Solving Nonlinear Systems of Equations”, Department of Applied Mathematics, University of Campinas, 1994

[10] Matson, J., B. Barret e J. Mellichamp, “Software Development Cost Estimation Using Function Points”, 1994

[11] Mendes, Emilia; Mosley, Nile; Counsell, Steve; “Web Metrics – Estimating Design and Authoring Effort”, IEEE Multimedia, Janeiro-Março, 2001

[12] Moores,T.T.; “Developing a software size model for rule-based system: a case study”, Expert system with applications,2001

[13] PMBOK, “A guide to the Project Management Body of Knowledge”, 2000 Edition

[14] Powell, T.A., “Web Site Engineering”, Prentice-Hall, 1998.

[15] Pressman, R.S., “Software Engineering”, Fifth Edition, 2001

[16] Putnam, L H, “General empirical solution to the macro software sizing estimation problem”, IEEE Transactions on Software Engineering, Vol SE4 Nº4, 1978

[17] Reifer,Donald J. ; ”Web Development: Estimating Quick-to-Market Software”, IEEE Software, Novembro-Dezembro, 2000

[18] Symons, C. J.; “Software Sizing and Estimating: MkII Function Point Analysis”, Wiley and Sons, 1991

[19] Stat Soft,Inc. STATISTICA for Windows (Computer Program Manual), 1999.
Web: http://www.statsoft.com

[20] Telecom Glossary, 2000 http://www.atis.org/tg2k/_world_wide_Web.html, acesso em 2003