FERNANDO DE MATOS MÔRA

MODELOS DE ESTIMATIVAS DE SOFTWARE BASEADOS EM DADOS HISTÓRICOS

Londrina

2009

FERNANDO DE MATOS MÔRA

MODELOS DE ESTIMATIVAS DE SOFTWARE BASEADOS EM DADOS HISTÓRICOS

Monografia apresentada ao Curso de Especialização em Engenharia de Software e Banco de Dados da Universidade Estadual de Londrina, como requisito parcial à obtenção do grau de especialista em Engenharia de Software e Banco de Dados.

Orientador:
Prof. Ms. Rodolfo Miranda de Barros

Londrina
2009

MODELOS DE ESTIMATIVAS DE SOFTWARE BASEADOS EM DADOS HISTÓRICOS

FERNANDO DE MATOS MÔRA

Trabalho apresentado e aprovado com conceito ___ em ____ de ____ de 2009, pela Banca Examinadora constituída por:

_____

_____

_____

UEL
Londrina – PR
2009

AGRADECIMENTOS

Meus sinceros agradecimentos ao Mestre Rodolfo Miranda de Barros, pelo total apoio e comprometimento com a conclusão deste trabalho.

EPÍGRAFE

“[...] Para realizarmos coisas grandiosas, devemos sonhar tanto quanto agir *[...]”*

Anônimo

RESUMO

Este trabalho abordará sobre os modelos de estimativas de software baseados em dados históricos. Coletar dados, calcular e analisar métricas são três passos que devem ser implementados para iniciar um programa de métricas. A estimativa de projetos de software não é uma ciência exata, sendo apenas uma combinação de bons dados históricos e técnicas sistemáticas podendo melhorar a precisão da estimativa. Utilizando dados históricos, o FP (*Function Point*) pode então ser usado para (1) estimular o custo ou esforço necessário para projetar, codificar e testar o software; prever o número de erros que vão ser encontrados durante o teste; e (3) prever o número de componentes e/ou o número de linhas de código projetadas no sistema implementado.

ABSTRACT

This work will approach on the models of based estimates of software in historical data. To collect data, to calculate and to analyze metric are three steps that must be implemented to initiate a program of metric. The estimate of software projects is not an accurate science, being only one systematic combination of good historical data and techniques being able to improve the precision of the estimate. Using given historical, FP (Function Point) can then be used for (1) stimulating the cost or effort necessary to project, to codify and to test software; to foresee the number of errors that go to be found during the test; e (3) to foresee the number of components and/or the projected line number of code in the implemented system.

SUMÁRIO

1 INTRODUÇÃO

Boa parte dos problemas que ocorre durante o desenvolvimento de softwares é produzida pela falta de um processo de desenvolvimento do software bem definido, sendo causa de erros de estimativas de prazos e custos, falta de controle do processo, menor qualidade do produto final, pouca produtividade, carência de um processo repetível e diversos outros problemas.

Em projetos de desenvolvimento de softwares, pode-se afirmar que no presente, as estimativas de tempo e custo são imprecisas. As estimativas costumam ser realizadas de acordo com a experiência e “achismo” do gerente de projetos ou da equipe de desenvolvedores. Existe uma dificuldade muito grande em realizar estas estimativas com precisão, pois nem sempre são utilizados métodos e técnicas eficientes neste processo.

Um dos aspectos cruciais do planejamento e gerenciamento de projetos é a compreensão de quanto o projeto provavelmente custará. Estimativas de custo elevadas podem fazer com que os clientes cancelem projetos, ao passo que uma estimativa de custo abaixo do real pode forçar a equipe do projeto a investir muito de seu tempo, sem conseguir compensação financeira (PFLEEGER, 2007).

Estimativa de tamanho de software é um processo pelo qual uma pessoa ou um grupo de pessoas estima o tamanho de um produto de software (MCPHEE, 1999, p.11).

1.1 Problema

Como estimar prazo e custo de projetos de desenvolvimento de software?

1.2 Objetivos

1.2.1 Objetivos Gerais

Deseja-se propor um modelo de estimativa de tempo e custo para projeto de desenvolvimento de software.

1.2.2 Objetivos Específicos

Deseja-se investigar tipos de métricas de softwares que possam ser aplicadas em projetos de desenvolvimento de software. Realizar um estudo de caso onde o projeto tenha extrapolado o prazo ou os custos e aplicar uma métrica de software neste projeto.

1.3 Justificativa

As métricas de softwares são medidas de alguma propriedade de parte do software ou de sua especificação. Desde que os métodos quantitativos provaram ser eficientes em outras ciências, profissionais da ciência da computação tem trabalhado arduamente para trazer os mesmos benefícios ao desenvolvimento de software. Tom DeMarco afirmou, “Não se pode controlar aquilo que não se pode medir”. Desta forma será realizado um estudo de métricas existentes que possam ser utilizadas...

1.4 Hipóteses

Se os modelos de estimativa forem utilizados não resolverá o problema de imediato, mas criará uma cultura na empresa. Ao longo do tempo poder-se-á utilizar o histórico de projetos passados para “calibrar” as estimativas futuras. (...)

1.5 Metodologia

O objetivo de uma investigação científica é chegar à veracidade dos fatos por meio de um método que permita atingir determinado assunto. Define-se método como "o caminho para se chegar a determinado fim. E método científico como o conjunto de procedimentos intelectuais e técnicos adotados para se atingir o conhecimento” (GIL, 1994).

Por isso esse estudo trata-se de uma pesquisa aplicada quanto à natureza do fenômeno (gera conhecimentos para aplicação prática) e explicativa quanto aos objetivos, e exploratória do ponto de vista dos procedimentos técnicos de coleta de dados (foram realizadas pesquisas com dados pertencentes a estudos

anteriores). As informações necessárias para suportar este estudo foram obtidas de consultas bibliográficas e documentais.

2 POR QUE MEDIR SOFTWARE?

2.1 Motivação

Segundo Vazquez *et al* (2003, p.17) a primeira pergunta que deve ser respondida ao apresentar a técnica da análise de pontos de função é: qual a motivação para sua utilização? É fundamental levar em consideração três principais aspectos que devem ser avaliados para respondê-la:

- o contexto de sua aplicação nas organizações que mantém projetos operações voltados à contratação, desenvolvimento e manutenção de sistemas.

- a análise do dilema presente nesse contexto.

- o entendimento de como a técnica pode ajustar essas organizações a identificar e equacionar o conjunto de questões envolvidas na solução das dificuldades inerentes a esses empreendimentos.

O contexto citado anteriormente é influenciado por duas forças que, apesar de antagônicas entre si, são mutuamente complementares. Por um lado existem os requisitos do sistema que tendem a aumentar com o tempo, com qualidade, funcionalidade e desempenho. Todos aqueles que vivenciaram essa situação sabem que mesmo após o fechamento da fase de levantamento de requisitos, quando todas as especificações funcionais e não-funcionais foram concluídas, à medida que o produto vai tomando forma, seus futuros usuários já idealizam novas características inicialmente não percebidas. Independentemente de o projeto atender a essas necessidades nessa versão ou em alguma próxima, de tal fato só vem por ratificar a natureza expansiva dos requisitos. Muitas vezes, com a utilização do produto, aqueles que eram desejáveis passam a ser necessário, e muitos outros, até então inexistentes, são percebidos.

2.2 Gerência de projetos

Gerência é uma palavra que deriva de gerir, originária do latim, *gerere* que significa trazer, produzir, criar, executar e administrar. Em resumo, gerir é

planejar, executar e controlar. De acordo com o *Project Management Institute* (PMI) um projeto é um empreendimento temporário posto em execução para criar um único produto ou serviço. Na condução desse empreendimento, existe inúmeros processo que visam permitir a sua gerencia. Esses processos são agrupados em alguns passos básicos, dos quais se destacam três:

- planejamento: os objetivos são definidos e refinados e o melhor curso de ação é selecionado;

- execução: coordenação de pessoas e outros recursos para executar o plano;

Controle: garantia de que os objetivos são alcançados, por meio do monitoramento e medição regular do progresso, identificado as variações no plano e tomando as ações corretivas conforme a necessidade.

A utilização desse enfoque no desenvolvimento de sistemas busca a definição e implementação de processos, por intermédio dos quais seja possível controlar sua execução pelo conhecimento prévio do efeito das suas respostas. Aliado ao uso de tecnologias e metodologias mais produtivas, é uma dos melhores instrumentos para alcançar uma melhor relação entre custo e benefício no desenvolvimento de sistemas.

2.3 Visibilidade

O mecanismo descrito anteriormente é chamado retroalimentação e a palavra-chave que define seu principal benefício é visibilidade. Com ela tanto é possível saber para onde ir antes de iniciar a caminhada quanto se o caminho em curso é o correto. Ou seja, permite definir objetivos realistas e a adoção de medidas necessárias à prevenção e correção dos desvios que causam impacto no cumprimento desses objetivos. Essas medidas são um dos objetivos dos processos de planejamento e controle do projeto.

2.4 Planejamento

O planejamento de projetos de sistemas envolve equacionar um conjunto complexo de elementos, entre eles:

- quais produtos devem ser desenvolvidos (especificações, manuais, programas, módulos, base de dados, planos de implantação e conversão, etc.;

- por meio de que atividades (identificação de classes de objetos, preparação de cenários de interações típicas, construção de modelo funcional, validação de documentação técnica, etc.);

- de que profissionais (analistas, implementadores, documentadores, usuários, etc.);

- durante quanto tempo (semanas, meses, anos, etc.);

- quais os riscos devem ser identificados e contigenciados.

A equipe envolvida no planejamento do projeto, antes de seu início, normalmente possui apenas duas pessoas fontes de informação para ponderar essas variáveis:

- a data limite, estabelecida por fatores e pressões externas, o orçamento e o tempo disponível dos profissionais da empresa.

- a experiência adquirida individualmente pelos analistas e gerentes responsáveis pelo planejamento do projeto.

2.5 Controle

O controle é uma das principais atividades envolvidas na gerência de projetos. Trazendo estes conceitos para o contexto de um projeto de desenvolvimento de sistemas, é possível ter algumas idéias interessantes na busca da resposta de por que medir. Afinal, não se consegue controlar o que não se consegue medir.

2.6 O que medir?

Segundo Fenton e Pfleeger (1997), medição é o processo de obtenção de uma medida para uma entidade real. Uma medida fornece uma indicação de quantidade, dimensão, capacidade ou tamanho de algum produto de *software* ou de um processo. Em outras palavras, uma medida refere-se a um valor de uma métrica.

Segundo a norma ISO 9126 (2001), métrica é a composição de métodos para medição e escalas de medição.

Uma vez estabelecido por que medir, a próxima pergunta que se deve responder é: o que medir: a medida é, por definição, a quantificação de uma característica. No caso da área de sistemas devem ser avaliadas não só suas características de produto final, também as características dos processos envolvidos em sua concepção e construção. Dessa forma primeiro é preciso identificar as características relevantes para análise.

2.7 Qual medida melhor representa o tamanho?

Ao explorar a motivação e os objetos de medição, chega-se à conclusão que o tamanho é uma das propriedades que deve ser medida. Agora, é necessário avaliar a melhor unidade para medir o tamanho de sistemas.

A unidade mais intuitiva é o número de linhas de código. Os estudantes da área, ao iniciarem na prática de programação, ao comparem entre si os programas desenvolvidos para resolver um problema específico, já utilizam o numero de linhas de código como parâmetro de comparação.

Para Vazquez *et al* (2003, p.17) a aparente facilidade o uso de linhas de código (LOC) é perigosa. Pode-se incorrer no risco de usar dois pesos e duas medidas, se alguns pontos não forem esclarecidos ao exprimir quantidade de LOC, como, por exemplo:

- A inclusão na contagem da quantidade de linhas de comentários, linhas em branco ou comandos nulos;

- a inclusão na contagem de diretrizes de compilação;

- a contagem de múltiplos comandos ou declarações em uma única linha como varias linhas, uma para cada comendo ou declaração;

- a contagem de uma única linha nos casos em que um único comando ou declaração é expresso em múltiplas linhas;

- a inclusão na contagem de delimitadores de blocos de comandos nos casos em que de fato haja mais de um comando;

- a desconsideração de delimitadores de blocos de comandos nos caso em que sua utilização seja opcional.

Alguns reveses na utilização de LOC como unidade para medir o tamanho de sistemas:

- falta de padronização;

- falta de significado para os clientes usuários do objeto de medição;

- dificuldade de aplicação nas fases iniciais do ciclo de vida;

- dependência da linguagem de programação utilizada.

Exatamente pela urgência na solução desses reveses através de uma nova métrica complementar à contagem das Linhas de Código, em 1975, Allan Albrect definiu os conceitos que serviram de base para a métrica dos pontos de função (VAZQUEZ *et al*, 2003, p.34).

A técnica de analise de pontos de função é um método padrão para medir sistemas do ponto de vista de seus usuários pela quantificação da funcionalidade solicitada e fornecida.

2.8 Estabelecimento de um programa de métricas de software

Pressman (2006, p.514) escreve que, o software *Engineering Institute* desenvolveu um manual de diretrizes abrangente para estabelecer um programa de métricas de software “orientado a metas”, que sugere os seguintes passos:

1. Identifique suas metas de negocio.
2. Identifique o que você deseja saber ou aprender.
3. Identifique suas submetas.
4. Identifique as entidades e atributos relacionados a suas submetas.
5. Formalize suas metas de medição.
6. Identifique questões quantificáveis e os indicadores correlacionados que você vai usar para ajudá-lo a atingir suas metas de medição.

7. Identifique os elementos de dados que você vai coletar para construir os indicadores que ajudam a responder a suas perguntas.
8. Defina as medidas a ser usadas e torne essas definições operacionais.
9. Identifique as ações que você executará para implementar as medidas.
10. Prepare um plano para implementar as medidas.

De acordo com Pressman (2006, p.514), como o software apóia funções do negócio, diferencia sistemas ou produtos baseados em computador ou age como um produto em si mesmo, as metas definidas para o negócio podem, quase sempre, ser rastreadas até metas específicas de engenharia de software. Por exemplo, considere uma empresa que produz sistemas avançados de segurança residencial, que tem substancial conteúdo de software. Trabalhando em equipe, gerentes de engenharia de software e de negócio podem desenvolver uma lista de metas de negócio:

1. Aperfeiçoar a satisfação de nossos clientes com nossos produtos.
2. Tornar nossos produtos mais fáceis de usar.
3. Reduzir o tempo para colocação de um novo produto no mercado.
4. Facilitar o apoio aos nossos produtos.
5. Aperfeiçoar nossa lucratividade global.

Segundo Pressman (2006 515), deve ser considerado a quarta meta: “facilitar o apoio aos nossos produtos”. A seguinte lista de perguntas pode ser originada dessa meta:

- os pedidos de modificação do cliente contêm a informação de que precisamos para avaliar adequadamente a modificação e depois implementá-la a tempo?

- qual o tamanho da lista de pedidos de modificação pendentes?

- nosso tempo de resposta para corrigir defeitos é aceitável em relação ás necessidades do cliente?

- nosso processo de controle de modificação está sendo seguido?

As modificações com alta prioridade são implementadas a tempo?

Pressman (2006, p.515) comenta que:

> Com base nessas perguntas, a organização de software pode criar a seguinte submeta: aperfeiçoar o desempenho da gestão do processo de modificação. As entidades e atributos do processo de software, que são relevantes para a submeta, são identificados, e metas de medição associadas a eles são delineadas.

O SEI oferece diretrizes detalhadas para os passos 6 a 10 de sua abordagem de mediação orientada a metas. Em essência, um processo de refinamento passo a passo é aplicado, no qual as metas são refinadas até chegar a perguntas, que são então refinadas até chegar a entidades e atributos que são depois refinados até chegar a métricas (PRESSMAN, 2006, p.515).

3 ANÁLISE DE PONTOS DE FUNÇÃO

Segundo Garmus e Herron (2001), a APF é uma das abordagens mais utilizadas e estudadas atualmente.

A Análise de Pontos de Função (APF) é uma das métricas de estimativa de tamanho mais sedimentadas no mercado e que proporciona resultados cada vez mais precisos à medida que artefatos da fase de análise e projeto são gerados.

A APF mede o tamanho do *software* pela quantificação de suas funcionalidades, com base no projeto lógico ou a partir do modelo de dados, segundo a visão e os requisitos do usuário final (IFPUG, 2004).

Análise de Pontos de Função-APF, é uma técnica de medição das funcionalidades fornecidas por um software do ponto de seu usuário. Ponto de função é a unidade de medida desta técnica que tem por objetivo tornar a medição independente da tecnologia utilizada para a construção do software. Ou seja, a APF busca medir o que o software faz, e não como ele foi construído (VAZQUEZ *et al*, 2003, p.35).

De acordo com Azevedo (2003) a análise por pontos de função auxilia: no cálculo do custo real do software, na estimativa do custo do projeto e da implementação, no entendimento dos gastos na manutenção, nas negociações de contrato entre outros.

Segundo Hazan (2001):

> A Análise de Pontos de Função (APF) é um método-padrão para a medição do desenvolvimento de software, tendo em vista o estabelecimento de uma medida de tamanho do software em Pontos de Função (PFs), com base na funcionalidade a ser implementada, sob o ponto de vista do usuário.

O processo de medição, ou a contagem de pontos de função, é baseado em uma avaliação padronizada dos requisitos lógicos do usuário (VAZQUEZ *et al*, 2003, p.35).

Empiricamente as principais técnicas de estimativa de projetos de desenvolvimento de software assumem que o tamanho de um software é um vetor importante para a determinação do esforço para sua construção. Logo, saber o seu

tamanho e um dos primeiros passos do processo de estimativa de esforço, prazo e custo (VAZQUEZ, 2003, p.35).

Os pontos de função não medem diretamente esforço, profundidade ou custo. É exclusivamente uma medida de tamanho funcional do software. Este tamanho em conjunto com outras variáveis é que poderá ser usado para derivar profundidade, estimar esforço e custo.

Os seguintes passos devem ser observados para mensuração de tamanho do software utilizando esta abordagem (IFPUG, 2004):

> I - Estabelecer o objeto da contagem (projetos de desenvolvimento, projetos de manutenção ou contagem de uma aplicação);
>
> II - Determinar a fronteira de automação (deve ser sempre determinada sob o ponto de vista do usuário);
>
> III - Contar as funções de dados, reconhecidas pelo usuário, classificados em Arquivos Lógicos Internos (ALIs - que são grupos lógicos de dados mantidos dentro da fronteira da aplicação) e Arquivos de Interface Externa (AIEs – arquivos somente referenciados pela aplicação e mantidos dentro de outra fronteira de aplicação);
>
> IV - Contar as funções transacionais, divididos em Entradas Externas (EEs), Saídas Externas (SEs) e Consultas Externas (CEs);
>
> V - Calcular pontos de função não ajustados – a quantidade de funções de dados e transações com seus respectivos níveis de complexidade2 é multiplicada pelos números de pontos de função definidos em tabelas;
>
> VI - Determinar o Fator de Ajuste (conjunto de 14 CGS que influenciarão a complexidade do software. São elas: comunicação de dados, processamento distribuído, performance, utilização de equipamentos, volume de transações, entrada de dados on-line, eficiência do usuário final, atualização on-line, processamento complexo, reutilização de código, facilidade de implantação, facilidade operacional, processamento distribuído, facilidade de mudanças). A determinação do Fator de Ajuste considera a avaliação de cada característica numa escala de 0 (nenhuma influência) a 5 (grande influência). A determinação deste Fator de Ajuste (FA) é baseada na equação: FA = 0,65 + (0,01 x Soma das CGS); e
>
> VII - Determinar o tamanho do projeto (considera as funções de dados, transacionais, fatores de ajuste e tipo de projeto).

De acordo com Hazan (2001), o procedimento para contagem de PFs compreende sete passos:

> Determinar o tipo de contagem de pontos de função: este é o primeiro passo para o processo de contagem, sendo que existem três tipos de contagem: contagem de PF de projeto de desenvolvimento, de aplicações instaladas e de projetos de manutenção.
>
> Identificar o escopo de contagem e a fronteira da aplicação: neste passo, definem-se as funcionalidades que serão incluídas em uma contagem de PFs especifica. A fronteira da aplicação é definida estabelecendo um limite lógico entre a aplicação que está sendo medida, o usuário e outras

aplicações. O escopo de contagem define parte do sistema (funcionalidades) a ser contada.

Determinar a contagem de pontos de função não ajustados: os pontos de função não ajustados (PFNA) refletem as funcionalidades fornecidas pelo sistema para o usuário. Essa contagem leva em conta dois tipos de função: e dados e transacionais, bem como sua complexidade (simples, média ou complexa).

Contagem das funções de dados: as funções representam as funcionalidades relativas aos requisitos de dados internos e externos à aplicação. São elas os arquivos lógicos internos e os arquivos de interface externa. Ambos são grupos de dados logicamente relacionados ou informações de controle que foram identificados pelo usuário. A diferença está no fato de um Arquivo Lógico Interno (ALI) ser mantido dentro da fronteira da aplicação, isto é, armazenar os dados mantidos através de um ou mais processos elementares da aplicação, enquanto que um Arquivo de Interface Externa (AIE) é apenas referenciado pela aplicação ou, seja ele é mantido dentro da fronteira de outra aplicação. Assim, o objetivo de um AIE é armazenar os dados referenciados por um ou mais processos elementares da aplicação sendo contada, mas que são mantidos por outras aplicações. O objetivo principal de uma EE é manter um ou mais ALIs ou alterar o comportamento do sistema. As Saídas Externas (SEs) são processos elementares que enviam dados (ou informações de controle) para fora da fronteira da aplicação, mas sem realização de nenhum cálculo nem a criação de dados derivados. Seu objetivo e apresentar informação para o usuário, por meio de uma recuperação das informações. Nenhum ALI é mantido durante a realização, nem o comportamento do sistema é alterado.

Contagem das funções transacionais: as funções transacionais representam as funcionalidades de processamento de dados do sistema fornecidas para o usuário.

Determinar o valor do fator de ajuste: o fator de ajuste é baseado em 14 características gerais de sistemas, que avaliam a funcionalidade geral da aplicação que está sendo contada, e seus níveis de influência. O nível de influencia de uma característica é determinado com base em uma escala de 0 (nenhuma influência) a 5 (forte influência).

Calcular os pontos de função ajustados: finalmente, os PFs ajustados são calculados, considerando-se o tipo de contagem definido no primeiro passo.

3.1 Objetivos

Para Hazan (2001) os objetivos da APF são:

- medir as funcionalidades do sistema requisitadas e recebidas pelo usuário;

- medir projetos de desenvolvimento e manutenção de software, sem se preocupar com a tecnologia que será utilizada na implementação.

3.2 Conceito de usuário

Usuário tem um conceito mais amplo para a Análise de Pontos de Função do que o usual. O conceito não será restrito apenas a pessoa física que usa o software. Deve se entender por usuário como qualquer pessoa ou coisa que interaja com a aplicação ou especifique seus requisitos. Exemplo de usuário: pessoa física, uma outra aplicação, um hardware, um ator em um caso de uso, agentes governamentais (exigências regulatórias do governos normalmente abrangem boa parte de um software), gestores do negócio que o software irá atender.

3.3 Propósito da contagem de pontos de função

De acordo com Carlos Eduardo Vazquez *et al* (2003), uma contagem de pontos de função não é um fim em si mesmo, sempre há uma motivação maior – o seu propósito. Por exemplo: contagem com o propósito de medir o serviço entregue por um fornecedor para sua posterior remuneração. Ou ainda: contagem com o propósito de fornecer elementos para uma estimativa de custo de um projeto de software. De acordo com essa motivação é possível assumir algumas premissas que podem agilizar o processo. Logo, é possível efetuar contagens com níveis diferenciados de detalhe e também de precisão. Sendo mais objetivo, o propósito da contagem de pontos de função é fornecer uma resposta a um problema de negocio. O propósito:

- determina algumas premissas para o processo de contagem;

- determina o tipo de contagem: projeto de desenvolvimento, projeto de melhoria ou aplicação.

- determina o escopo da contagem, ou seja, se ela abrangerá um ou mais aplicações ou então apenas parte de uma aplicação.

- afeta o posicionamento da fronteira da aplicação. Por exemplo, a organização decide comprar um pacote de mercado com um módulo de seu sistema corporativo, com uma fronteira própria.

Determina o nível de detalhe da contagem. Por exemplo, se uma contagem está sujeita a uma auditoria posterior, é necessário que cada estapa do processo esteja bem documentada para facilitar a tarefa.

3.4 Determinação do tipo de contagem

Em determinar o tipo de contagem os responsáveis pela medição estabelecem o tipo de contagem que será utilizado para medir o software. A contagem de pontos de função pode estar associada tanto a projetos quanto a aplicações.

Os três tipos de contagem são os seguintes:

Contagem de um projeto de desenvolvimento: o numero de pontos de função de um projeto de desenvolvimento mede a funcionalidade fornecida aos usuários finais do software quando da sua primeira instalação. Isso significa que essa contagem também abrange as eventuais funções de conversão de dados necessárias à implantação do sistema.

- contagem de um projeto de melhoria: o número de pontos de função de um projeto de melhoria mede as funções adicionadas, modificadas ou excluídas do sistema pelo projeto, e também as eventuais funções de conversão de dados.

- contagem de uma aplicação (ou *baseline*): o número de pontos de função de uma aplicação mede a funcionalidade fornecida aos usuários por uma aplicação instalada. Também é chamado de número de pontos de função instalados ou *baseline*. Esse número fornece uma medida da atual funcionalidade da aplicação.

3.5 Funções do tipo dado

Segundo Vazquez *et al* (2003), as funções do tipo dados representam as funcionalidades fornecidas pelo sistema ao usuário, para atender a suas necessidades de dados. São classificadas em:

Arquivo Lógico (ALI): um grupo logicamente relacionado de dados ou informações de controle, identificável pelo usuário, mantido dentro da fronteira da aplicação sendo contada. Sua intenção é armazenar dados mantidos através de uma ou mais transações da aplicação sendo contada.

Arquivo de Interface Externa (AIE): um grupo logicamente relacionado de dados ou informações de controle, identificável pelo usuário, mantidos fora da fronteira da aplicação sendo contada.

3.6 Funções do Tipo Transação

Vazquez *et al* (2003, p.60) escreve que as funções do tipo transação representam as funcionalidades de processamento de dados fornecidas pelo sistema ao usuário. São classificadas em:

> Entrada Externa (EE): é uma transação que processa dados ou informações de controle originados de fora da fronteira da aplicação. Sua principal intenção é manter um ou mais arquivos lógicos internos e/ou alterar o comportamento do sistema. Exemplo: incluir cliente, alterar cliente, excluir cliente.
>
> Saída Externa (SE): é uma transação que envia dados ou informações de controle para fora da fronteira da aplicação. Sua principal intenção é apresentar informação ao usuário através de lógica de processamento
>
> Consulta Externa (CE): é uma transação que envia dados ou informações de controle para fora da fronteira da aplicação. Sua principal intenção é apresentar informações ao usuário através da simples recuperação de dados ou informações de controle de ALIs e/ou AIEs. Exemplo: consulta cadastro de clientes.

Conforme Vazquez *et al* (2003) as funções do tipo transação representam a funcionalidade fornecida ao usuário para atender as suas necessidades de processamento de dados pela aplicação. São classificadas em Entradas Externas (EE), Saídas Externas (SE) ou Consultas Externas (CE).

3.6.1 Entrada Externa

Silva (2000) afirma que uma Entrada Externa é um processo elementar que recebe dados vindos de fora da fronteira do aplicativo. Os dados podem ser recebidos de telas de entrada de dados ou diretamente de outros aplicativos. Os dados podem ser informações de negócio ou informações de controle. Uma informação de negócio é um dado que precisa ser tornado persistente, ou seja, que precisa ser armazenado em algum arquivo lógico interno. Já uma informação de controle é um dado que influência um processo elementar.

Segundo Vazquez *et al* (2003) uma definição para Entrada externa é:

- Um processo elementar;

- Que processa dados ou informações de controle recebidos de fora da fronteira da aplicação;

- Cuja principal intenção é manter um ou mais arquivos lógicos internos e ou modificar o comportamento do sistema.

3.6.2 Saída Externa

Silva (2001) entende que:

> Uma Saída Externa é um processo elementar através do qual dados derivados são enviados para além da fronteira do aplicativo. Os dados podem ser relatórios ou arquivos de saída que são enviados para outros aplicativos. A finalidade básica de uma saída externa é a de recuperar dados, processá-los e enviá-los para além da fronteira do aplicativo. No entanto, uma saída externa pode, às vezes, atualizar arquivos lógicos internos.

Vazquez *et al* (2003) uma definição para saída externa é:

- Um processo elementar;

- Que envia dados ou informações de controle para fora da fronteira da aplicação;

- Cuja principal intenção é apresentar informação ao usuário por meio de lógica de processamento que não seja apenas a recuperação de dados ou informações de controle.

3.6.3 Consulta Externa

Silva (2000) afirma que uma Consulta Externa é um processo elementar com componentes de entrada e saída recuperados de arquivos lógicos internos e de interface externa. Os dados são enviados para alem da fronteira da aplicação. O processo não atualiza nenhum arquivo lógico interno e os dados enviados não sofrem qualquer tipo de transformação.

Nas palavras de Vazquez *et al* (2003) uma definição para Consulta externa é:

- Um processo elementar;

- Que enviam dados ou informações de controle para fora da fronteira da aplicação;

- Cuja principal intenção é apresentar informação ao usuário por meio de uma simples recuperação de dados ou informações de controle de um ALI ou AIE.

3.7 Pontos de Função Não-Ajustados

Cada função do tipo e transação é classificada com relação à sua complexidade em baixa, média e alta. A complexidade das funções do tipo dado é determinada pela quantidade de tipos de dados (campos) e tipos de registro (subgrupos de dados dentro do arquivo). As funções do tipo transação têm sua complexidade determinada pela quantidade de tipos de dados e arquivos referenciados. A complexidade de cada tipo de função possui associado um valor em pontos de função não-ajustados. Por exemplo, um ALI de complexidade média possui dez pontos de função não-ajustados. Após a identificação e classificação de todas essas funções na contagem, o número de pontos de função não-ajustados será simplesmente a soma do valor de cada uma dessas funções (VAZQUEZ *et al* (2003, p.60).

3.8 Fator de Ajuste

Para Silva (2000) o fator de ajuste (VAF – *Value Adjustment Factor*) é baseado em 14 (quatorze) características gerais de sistema (CGS), listadas na seqüência:

1. Comunicação de Dados;
2. Performance;
3. Volume de Transações;
4. Interface com o Usuário;
5. Processamento Complexo;
6. Facilidade de Implantação;
7. Múltiplos Locais;

8. Processamento Distribuído;

9. Utilização de Equipamento;

10. Entrada de Dados "on-line";

11. Atualização "on-line";

12. Reutilização de Código;

13. Facilidade Operacional;

14. Facilidade de Mudanças.

Vazquez *et al* (2003) afirma que o IFPUG fornece diretrizes que ajudam a diminuir a subjetividade na determinação do nível de influência. O IFPUG diz que determinados os níveis de influência dos 14 CGS, o fator de ajuste é calculado da seguinte forma: VAF = (TDI x 0,01) + 0,65. Em que TDI (total *degree of influence*) é o somatório dos níveis de influência das características gerais. O fator de ajuste ajusta o número de pontos por função não-ajustados em no máximo + ou – 35%.

3.9 Pontos de Função Ajustados

Cada tipo de contagem (projeto de desenvolvimento, projeto de melhoria e aplicação) possui uma formula específica para a determinação dos pontos de função ajustados.

Segundo Hazan (2001), uma vez calculados os PF não ajustados e o fator de ajuste, é possível calcular os PFs ajustados. Esse cálculo é feito de formas diferentes para cada tipo de contagem (projeto de desenvolvimento, projeto de manutenção ou aplicações instaladas). Para projetos de desenvolvimento, o cálculo é dado por:

PF = PFNA * VFA

onde PFNA = Número de PFs não ajustados e

VFA = valor do fator de ajuste

4 ESTIMATIVA

Desde a década de 80, várias abordagens têm sido propostas e aperfeiçoadas com o objetivo de estimar funcionalmente o tamanho de um *software*, entre elas destaca-se a Análise por Pontos de Função - APF, proposta em 1979 e atualmente na versão 4.2 (IFPUG, 2004).

Ao identificar a necessidade de desenvolvimento, manutenção ou mesmo aquisição e customização de um software para atender a novas demandas de uma organização, dois questionamentos sempre presentes são os seguintes:

1. Quanto tempo será necessário para concluir o projeto?
2. Quanto o projeto vai custar para a empresa?

As particularidades dos requisitos de um projeto de software, bem como da equipe responsável e da tecnologia empregada, são fatores que dificultam a obtenção de respostas confiáveis a estas perguntas. Tais dificuldades podem ser evidenciadas ao tentar analisar questões como:

> - os requisitos traduzem com fidelidade as necessidades do negócio dos usuários? Já se encontram suficientemente estabilizados? Refletem características de software transacionais de baixa complexidade ou possuem atributos que exigem conhecimentos específicos, tais como alta performance, matemática complexa, processamento distribuído, etc.?
>
> - a equipe de desenvolvimento possui conhecimento na área de negócio que será atendida pelo projeto de software? Possui experiência na utilização das ferramentas necessárias à conclusão do projeto? Possui todos os perfis necessários impostos pelas características do projeto? Existem conflitos internos à equipe que precisam ser solucionados? É possível identificar uma liderança entre os integrantes da equipe? Qual o grau de motivação da equipe mediante o projeto?
>
> - a tecnologia já faz parte da cultura da organização? Pode ser facilmente absorvida por novos integrantes da equipe de projeto? Existe material de apoio suficiente para o aprendizado da tecnologia? Sua aplicação exige pessoal com algum tipo de especificação? Suporta a implementação de todas as características do software? Atende aos requisitos não-funcionais do projeto?

Mesmo quando tais questionamentos são realizados na área de engenharia civil, na produção de algo bem mais tangível que um software, como uma casa ou uma ponte, características específicas que diferenciam um novo projeto dos anteriores adicionam um certo grau de incerteza às respostas.

De acordo com Carlos Eduardo Vazquez *et al* (2003, p.156) o processo de estimativa de um projeto envolve, basicamente, quatro atividades:

1. Estimar o tamanho do produto a ser gerado;
2. Estimar o esforço empregado na execução do projeto;
3. Estimar a duração do projeto;
4. Estimar o custo do projeto.

Para obter as respostas aos questionamentos iniciais de tempo e custo a partir da aplicação de um processo de estimativa, antes de qualquer coisa, é necessário decidir a unidade para medir o tamanho do produto.

Devido ao propósito principal de um processo de estimativa ser o de fornecer informações que beneficiem o planejamento e o controle dos projetos de software o mais cedo possível durante seu ciclo de vida, a utilização de pontos de função como unidade padrão de tamanho é a mais acertada. Isso se dá por uma série de motivos.

Carlos Eduardo Vazquez *et al* (2003) escreve que:

> Em primeiro lugar, para que uma estimativa de tamanho baseada em LOC seja confiável, é necessário, atrasando a geração de indicadores úteis para o seu gerenciamento. A própria definição de linguagem de programação a ser utilizada no desenvolvimento do projeto, fundamental para o processo de contagem de linhas de código, nem sempre é realizada na fase inicial de seu ciclo de vida, já que algumas características funcionais e até não-funcionais do software devem ser exploradas antes de se tomar alguma decisão. Por outro lado, como a obtenção de pontos de função depende unicamente do conhecimento da funcionalidade requerida para o software e não da tecnologia empregada em seu desenvolvimento, uma estimativa de tamanho pode ser realizada desde a fase inicial do levantamento de requisitos e ser refinada à medida que se avança no entendimento do projeto.

É fato que softwares bem projetados ou com alto índice de componentização ou de reutilização de código têm seus indicadores baseados em LOC penalizados. Por exemplo, um sistema com 50.000 LOC, mas que exigiu um alto esforço para a definição de uma arquitetura bem elaborada, irá apresentar uma profundidade final média inferior à de outro projeto similar e de funcionalidade equivalente, porém com arquitetura menos trabalhada, com 60.000 LOC e que

pontos de função, essa distorção não seria observada, uma vez que não existiria diferença relevante no tamanho estimado de ambos os projetos.

O processo de estimativa inicia-se a partir dos primeiros níveis de abstração dos requisitos do projeto. Para ser completo e eficiente, deve em consideração as experiências e os dados históricos de projetos passados, os recursos que cercam os projetos. A cada etapa realizada, as estimativas obtidas devem passar por um processo de aprovação antes de registradas na base histórica. Além de poderem ser utilizadas como fonte de informação para projetos futuros, servirão como insumo para as estimativas das etapas seguintes. Reestimar deve ser uma atividade constante durante todo o ciclo de vida do desenvolvimento do software. Na etapa final do processo, com o produto concluído, as medidas reais de tamanho, histórica, servindo também para a validação e o aprimoramento de todo o processo de estimativa.

Independentemente da etapa em que se esteja no processo, existem técnicas de estimativas que podem ser utilizadas. Basicamente, essas técnicas são classificadas em duas categorias: métodos diretos ou métodos derivados.

Os métodos diretos são aqueles baseados na opinião de especialistas. O procedimento padrão consiste em reunir a opinião de um ou mais especialistas, que fornecem uma suposição direta da estimativa, buscando-se principalmente em suas intuições e experiências passadas.

Os métodos derivados, também conhecidos como métodos de modelo algorítmico ou métodos paramétricos, fornecem um ou mais algoritmos de transformação que produzem a estimativa desejada em função de variáveis que se relacionam com alguns atributos de software. Por exemplo, uma estimativa de tamanho não é realizada diretamente em valores de pontos de função, mas sim em diferentes atributos de software que podem ser correlacionados a eles.

4.1 Estimativa de tamanho em ponto de função

De acordo com IFPUG (1991), determinam-se os Pontos por Função de uma aplicação em três etapas de avaliação. A primeira resulta na contagem de pontos de funções não-ajustados, que refletem as funções específicas e mensuráveis do negócio, provida ao usuário pela aplicação. A segunda etapa da

avaliação gera o fator de ajuste, que representa a funcionalidade geral provida ao usuário pela aplicação. A terceira etapa resulta na contagem de Pontos por Função ajustados, que reflete o fator de ajuste aplicado ao resultado apurado na primeira etapa.

O primeiro passo para alcançar estimativas efetivas do projeto de software é estimar o seu tamanho.

Uma técnica simples e rápida para alcançar uma estimativa de tamanho, classificada como um método direto, é a analogia simples. A técnica de analogia é aplicada comparando o projeto atual com dados de projetos similares passados, obtidos de uma base histórica. A partir do conhecimento do tamanho de um projeto similar, estima-se o tamanho do novo projeto como um percentual do tamanho daquele, podendo-se até subdividir o projeto em módulos menores (de mais fácil compreensão) e estimar o percentual do tamanho de cada módulo em relação ao projeto passado. Então, o tamanho total do projeto é obtido pela soma dos tamanhos estimados para cada módulo separadamente. Logicamente, a eficácia desse método irá depender de dois fatores: a acurácia do valor do tamanho do projeto da base histórica e o grau de similaridade entre o novo projeto e seu antecessor.

Contudo as técnicas mais utilizadas para a estimativa de tamanho são encontradas na categoria dos métodos. Alguns dessas técnicas são tratados em seguida.

4.2 Contagem dedutiva

Essa técnica considera a possibilidade de um único componente da análise de pontos de função para a aplicação, geralmente o número de Arquivos Lógicos Internos, derivando o resto da contagem a partir de bases estatísticas.

No trabalho *Early Function Point Couting*, publicado pela NESMA, encontram-se descrita uma abordagem para este método. Trata-se da Contagem Indicativa, na qual apenas é necessária a identificação dos Arquivos de Interface

Externa (AIE) e Arquivos Lógicos Internos (ALI). Considera-se 35 PF para cada ALI e 15 PF para cada AIE identificado.

Os números 35 e 15 representam as médias de pontos de função identificadas para cada um dos tipos de arquivo, considerando projetos que, em média, possuam três Entradas Externas (EE), duas saídas Externas (SE) e uma Consulta Externa (CE) para cada ALI e uma Saída Externa e uma Consulta Externa para cada AIE.

4.3 Complexidade Média

A versão 5 do ISBSG Benchmark apresenta uma média de complexidade para os projetos de desenvolvimento, distribuídas por tipos de funções, conforme a tabela seguinte:

Tipo de Função	Média de PF	IFPUG (baixa)	IFPUG (média)	IFPUG (alta)
ALI	7,4	7	10	15
AIE	5,5	5	7	10
EE	4,3	3	4	6
SE	5,4	4	5	7
CE	3,8	3	4	6

Tabela de média de PF não-ajustados, por tipo de função, do ISBSG comparado na tabela de complexidade do IFPUG.

Para o processo de estimativa, após a identificação do número de todos os componentes funcionais (ALI, AIE, EE, SE e CE), basta que sejam associadas suas respectivas complexidades. O número estimado para o tamanho funcional é alcançado pela fórmula:

e =#EE x 4,3+#SE x 5,4+#CE x 3,8+# ALI x 7,4+# AIE x 5,5

A NESMA simplificou esta abordagem com sua Contagem Estimativa. Após a identificação de todas as funcionalidades do software, utiliza-se a classificação de complexidades do IFPUG e aplica-se a complexidade baixa para cada Arquivo Lógico Interno e Arquivo de Interface Externa e média para cada

Entrada Externa, Saída Externa ou Consulta Externa. A estimativa do tamanho é então obtida pela formula:

e = #EEx4+#SEx5+#CEx4+#ALIx7+#AIEx5

Tanto a Contagem Dedutiva quanto a técnica de Complexidade Média são baseadas em pontos de função não-ajustados para a obtenção das estimativas, uma vez que o fator de ajuste pode ser determinado sem muita dificuldade mesmo em estagio iniciais do projeto.

4.4 Backfiring

Esse método consiste em derivar o número de pontos de função da aplicação a partir de seu tamanho físico, medido em linhas de código, utilizando um fator de conversão constante dependente da linguagem de programação. A idéia em si possui bastante apelo, uma vez que a contagem de linhas de código pode ser feita por meio de ferramentas automáticas e, conseqüentemente, o numero de pontos de função poderia ser derivado de imediato. Por exemplo, utilizando um fator de conversão de 80 LOC/PF para C++ e tendo uma aplicação escrita em C++ com 8.000 linhas de código, chega-se a 1.000 pontos de função para essa mesma aplicação.

Entretanto, não raro essa técnica apresenta erros consideráveis quando confrontada com uma aplicação. Isso porque assume uma relação linear entre o tamanho físico, que são grandezas distintas alguns autores sugerem que se houvessem essa relação entre pontos de função e linhas de código, uma das seguintes assertivas deveria ser verdadeira:

- LOC não é uma medida técnica, mas fundacional;

- FP não é uma medida funcional, mas técnica;

- Medidas técnicas também são medidas funcionais.

Além disso, as variações encontradas nos próprios fatores de conversão publicados por inúmeras organizações no mercado podem chegar a 100%! A tabela seguinte apresenta um exemplo dessas variações para a linguagem COBOL.

Fonte	**Fator de Conversão (COBOL)**
Quantitative Software Management	77 LOC/PF
Capers Jones	107 LOC/PF
David Consulting Group	177 LOC/PF

Tabela de fatores de conversão obtidos do mercado para *backfiring em aplicações COBOL*.

Devido a esses fatores, aplicar *backfiring* para obter um tamanho em pontos de função a partir de linhas de código é uma técnica arriscada e sujeita a uma grande margem de erro. Daí, o próprio IFPUG ressalta no FAQ em sua página na Internet, contagem manual de pontos é inviável na prática e a precisão não seja um fator crítico. Alguns profissionais advogam que *backfiring* é uma maneira rápida e barata de obter o tamanho em pontos de função do portfólio de aplicações de uma organização. Outros argumentam que o barato sai caro; é melhor investir na contagem manual dos pontos de função e ter confiabilidade desses dados, com compensação no longo prazo.

Por outro lado, muitos modelos de estimativa de software, como o COCOO II, utilizam como dado primário de entrada de seu processo o tamanho em linha de código. Nesses casos é muito comum realizar o inverso: obter o número de linhas de código a partir do tamanho em pontos de função. Isso porque nas fases iniciais de um projeto de software é mais fácil estimar ou medir o seu tamanho em pontos de função do que em linhas de código. Mesmo nesse caso, as considerações anteriores sobre *backfiring* continuam válidas.

4.5 A Escolha do Tipo de Contagem

Logicamente, a contagem detalhada de pontos de função é muito mais precisa que os demais tipos de contagem na determinação do tamanho de um projeto. Contudo, esse tipo de contagem também requer um maior tempo para ser realizado e depende de especificações mais detalhadas a respeito dos requisitos do projeto. Ao contrário, uma contagem indicativa, por exemplo, pode ser realizada já nas fases iniciais do desenvolvimento, quando existe apenas uma visão superficial do projeto, com um bom grau de fidelidade de resultado.

4.6 Fato de Crescimento

Com a sistematização do dimensionamento de projetos nas várias fases do ciclo de vida de desenvolvimento e manutenção de sistemas, a organização passa a poder se favorecer de outro indicador muito importante no processo de estimativa: a variação no tamanho do projeto. Além de contribuir para a melhoria contínua do processo de desenvolvimento utilizado na organização, identificando fatores inerentes ao processo que contribuem para esse aumento de tamanho, o conhecimento de fator de crescimento (FC) dos projetos pode contribuir para a antecipação do crescimento do orçamento inicialmente definido para o projeto.

Para fins de simplificação de nossa discussão até o momento, consideramos que não há variação entre o momento da estimativa e da medição do tamanho do projeto. Na prática, contudo, essa simplificação é falsa. É regra, ao final dos projetos, serem apuradas mais funcionalidades que a estimada originalmente.

4.7 Estimativa de Esforço

No geral, o dimensionamento ou a estimativa da quantidade de pontos de função nos vários estágios do projeto, isoladamente, são relevantes como base para a remuneração de contratos. Os pontos de função passam a ter maior significado quando utilizados como parâmetros na obtenção de estimativas de outras variáveis relevantes para a gerência de projetos entre elas o esforço.

Ao utilizar a quantidade de pontos de função como base para a obtenção do esforço, considera-se a existência de alguma função que relacione

essas duas dimensões. Dentre as dificuldades para sua identificação, a principal, em que as particularidades de todas as etapas do ciclo de vida do desenvolvimento são conhecidas, bem como todos os subprodutos gerados em cada uma dessas etapas. Esse assunto é bastante abrangente é foge do escopo deste texto abordá-lo. Outra finalidade é a falta de cultura e experiência prática na aplicação de pontos de função, aliada a uma mitologia simplista que promete com uma simples multiplicação de alguns números mágicos fornecerem resultados próximos à realidade.

Por estes motivos geralmente se estima o esforço total de um projeto pela soma das estimativas de esforço de cada atividade. O principal insumo nesse processo e a experiência individual dos responsáveis pela estimativa. Uma prática tão comum observada nas organizações para alcançar uma estimativa de esforço é a utilização da (*Work Breakdown Structure – WBS*), em que o projeto de software é decomposto em suas funções e uma estimativa de esforço é fornecida para cada uma delas. Também é comum a utilização de métodos diretos, como as técnicas Delphi e *Three-Point*.

4.8 Estimativa de duração

O terceiro passo da estimativa de um projeto de software é determinar o cronograma do projeto a partir do esforço estimado. Dentre o planejamento do desenvolvimento ou manutenção de sistemas, a estimativa de sua duração é uma das atividades que exigem maior experiência e conhecimento dos aspectos envolvidos em sua execução. Pela utilização intensiva de trabalho envolvido e os recursos disponíveis para sua execução. Uma vez determinado o esforço em horas, necessário para a realização de uma determinada atividade, para obter sua estimativa de duração, basta dividir esse valor pelo número de horas trabalhadas pela equipe alocada.

$$\text{Prazo} = \frac{\text{esforço}}{\text{recursos}}$$

O prazo previsto será o prazo entre o esforço previsto e a quantidade de recursos alocada na execução do projeto. Tal proposição, que implica um uma relação linear entre as variáveis, é uma simplificação.

A experiência comprova que essa relação linear entre prazo e quantidade de recursos não existe, assim como não é verdade a afirmação de que “nove mulheres geram um bebê em um mês”. Para viabilizar a diminuição do prazo, também é adicionado esforço novo ao projeto. Esse trabalho é conseqüência de novas atividades que suprem as necessidades de comunicação, integração e seqüenciamento geradas pela maior distribuição das tarefas.

5 ESTIMATIVAS DE SOFTWARE BASEADOS EM DADOS HISTÓRICOS

5.1 Estimativa do projeto de software

O software é o elemento virtualmente mais caro de todos os sistemas baseados em computador. Para sistemas complexos, feitos sob medida, um erro de estimativa grande pode fazer a diferença entre lucro e prejuízo. Excesso de custo pode ser desastroso para o desenvolvedor.

Existem algumas opções para conseguir estimativas de custo e esforço confiáveis:

1. Adie a estimativa até que o projeto esteja mais adiantado (obviamente, podemos conseguir estimativas 100% precisas depois que o projeto estiver completo).
2. Baseie as estimativas em projetos semelhantes, que já foram completados.
3. Use técnicas de decomposição relativamente simples para gerar estimativas de custo e esforço do projeto.

Roger S. Pressman (2006, p.524) comenta que:

> Infelizmente, a primeira opção, apesar de atraente, não é prática. Estimativas de custo precisam ser fornecidas “logo”. Entretanto, é preciso reconhecer que quanto mais esperarmos, mais saberemos, e quanto mais soubermos, será menos provável cometermos erros sérios em nossas estimativas.

Infelizmente, a primeira opção, apesar de atraente, não é prática. Estimativas de custo precisam ser fornecidas “logo”. Entretanto, é preciso reconhecer que quanto mais esperarmos, mais saberemos, e quanto mais evidente, será menos provável o cometimento de erros sérios em nossas estimativas (PRESSMAN, 2006, p.524).

A segunda função pode funcionar razoavelmente bem se o presente projeto for bastante semelhante a esforços anteriores e as outras influências do projeto (por exemplo, o cliente, as condições do negocio, o SEE, os prazos) forem

equivalentes. Infelizmente, a experiência anterior nem sempre é um bom indicador de resultados futuros.

As opções restantes são abordagens viáveis para a estimativa de projetos de software. O ideal é que as técnicas citadas para cada opção sejam aplicadas em conjunto; cada uma usada para verificar a outra. Técnicas de decomposição usam a abordagem “dividir e conquistar” para a estimativa de projetos de software. Pela decomposição de um projeto em suas funções principais e atividades relacionadas de engenharia de software, a estimativa de custo e esforço pode ser realizada passo a passo. Modelos empíricos de estimativa podem ser usados para complementar as técnicas de decomposição e oferecem por si mesmo uma valiosa abordagem de estimativa (PRESSMAN, 2006, p.524).

Cada uma das opções viáveis de estimativa de custo de software é de boa na mesma medida em que o são dados históricos usados para alimentar a estimativa. Se não existem dados históricos, o custo se baseia em uma fundação precária.

A estimativa de projetos de software é uma forma de solução de problemas e, na maioria dos casos, o problema a ser resolvido (por exemplo, desenvolver uma estimativa de custo e esforço para um projeto de software) é muito complexo para ser considerado como um todo.

A estimativa usa uma ou ambas a formas de particionamento. Mas, antes que uma estimativa possa ser feita, o planejador do produto precisa entender o escopo do software a ser construído e gerar uma estimativa do seu “tamanho”.

5.2 Dimensionamento do Software

A precisão da estimativa de um projeto de software depende de alguns fatores:

I – o grau com que o planejador estimou adequadamente o tamanho a ser construído;

II – a aptidão para traduzir a estimativa de tamanho em esforço humano, tempo transcorrido e dinheiro (em função da disponibilidade de métricas de software confiáveis de projetos anteriores);

III – o grau com que o plano de projeto reflete a capacidade da equipe de software;

IV – a estabilidade dos requisitos do produto e do ambiente que apóiam o esforço de engenharia de software.

Putnam e Myers (1992) sugerem quatro abordagens diferentes para o problema do dimensionamento:

> - dimensionamento de “lógica nebulosa”. Para aplicar essa abordagem, o planejador deve identificar o tipo de aplicação, estabelecer em uma escala qualitativa e depois refiná-la dentro do intervalo original.
>
> - dimensionamento de pontos por função. O planejador desenvolve estimativas das características do domínio da informação.
>
> - dimensionamento de componentes padrão. O software é composto por vários “componentes padrão” diferentes, que são genéricos para uma área de aplicação específica. Por exemplo, os componentes padrão para um sistema de informação são subsistemas, módulos, telas, relatórios, programas interativos, programas com processamento por lotes, arquivos, LOC e instruções em nível de objeto. O planejador do projeto estima a quantidade de ocorrências de cada componente padrão e depois usa dados históricos de projeto para determinar o tamanho correspondente a cada componente padrão.
>
> - dimensionamento de modificação. Essa abordagem é usada quando um projeto abrange o uso de software existente, que precisa de algum modo como parte do projeto. O planejador estima a quantidade e o tipo (por exemplo, reuso, adição de código, modificação de código, eliminação de código) das modificações que precisam ser efetuadas.

Putnam e Myers sugerem que os resultados de cada uma dessas abordagens de dimensionamento sejam combinados estatisticamente para criar uma estimativa de três pontos ou valor esperado. Isso é conseguido desenvolvendo um valor otimista (baixo), um mais provável e um pessimista (alto) para o tamanho, e combinando-os por meio da equação.

Roger S. Pressman (2006, p.526) afirma que:

> Linhas de código e pontos por função foram descritos como medidas a partir das quais as métricas de profundidade podem ser calculadas. Dados de LOC e FP são usados de dois modos durante a estimativa de projetos de software: I – como variável de estimativa para “dimensionar” cada elemento do software; II – como métricas referenciais coletadas de projetos anteriores e usadas juntamente com variáveis de estimativa para desenvolver projeções de custo e esforço.

As estimativas LOC e FP são técnicas distintas. Todavia, ambas têm algumas características em comum. O planejador do projeto começa com uma declaração delimitada do escopo do software e a partir dela tenta decompor o software em funções do problema que podem ser estimadas individualmente. LOC ou FP (a variável de estimativa) é então estimada para cada função. Como alternativa, o planejador pode escolher outro computador para dimensionar, como classes ou objetos modificações ou processos do negocio afetados.

Métricas referenciais de profundidade (por exemplo, LOC/PM ou FP/pm[5]) são apenas à variável de estimativa adequada e o custo ou esforço correspondente à função é derivado. As estimativas de função são combinadas para produzir uma estimativa global para todo o projeto.

Freqüentemente há substancial espalhamento nas métricas de produtividade de uma organização, fazendo com que o uso ou FP/PM devem ser calculadas por domínio de projeto, isto é, os projetos devem ser agrupados por tamanho de equipe, área de aplicação, complexidade e outros parâmetros relevantes. Médias locais dos domínios devem então ser calculadas. Quando um novo projeto é estimado, deve ser primeiro classificado em um domínio e depois a média de produtividade do domínio adequado deve ser usada para gerar a estimativa.

As técnicas de estimativa LOC e FP diferem quanto ao nível de detalhe necessário para a decomposição e quanto ao objetivo do particionamento. Quando LOC é usada como variável de estimativa, a decomposição é absolutamente essencial e é freqüentemente levada a um considerável nível de detalhe. Quanto maior for o grau de particionamento, mais provável será que estimativas razoavelmente precisas de LOC possam ser desenvolvidas.

Para estimativas de FP, a decomposição funciona de modo diferente. Em vez de focalizar a função, cada uma das cinco características do domínio de informação bem como os 14 valores de ajuste de complexidade são estimados. As estimativas resultantes podem, então, ser usadas para derivar um valor de FP, que pode ser relacionado a dados anteriores e usado para gerar uma estimativa.

Independentemente da variável de estimativa que é usada, o planejador do projeto começa estimando um intervalo de valores para cada função ou valor do domínio da informação. Uma indicação implícita do grau de incerteza é fornecida quando um intervalo de valores é especificado.

Um valor esperado, ou de três pontos, pode ser calculado. O valor esperado para a variável de estimativa (tamanho), S, pode ser calculado como média ponderada das estimativas otimista (S_{ot}) mais provável (S_m) e pessimista (S_{pess}). Por exemplo,

$$S = (S_{ot}\ 4S_m + S_{pess})/6$$

dá maior peso à estimativa “mais provável” e segue uma distribuição de probabilidade beta. Consideramos que há uma probabilidade muito pequena de que o resultado correspondente ao tamanho real caia fora do intervalo entre os valores otimista e pessimista.

Uma vez determinado o valor esperado para a variável para a variável de estimativa, os dados históricos de produtividade LOC ou FP são aplicados. As estimativas estão corretas? A única resposta razoável para essa questão é: não podemos estar seguros. Qualquer técnica de estimativa, não importa quão sofisticada, deve ser comparada com outra abordagem. Ainda assim, devem prevalecer o bom senso e a experiência.

5.3 Um exemplo de estimativa baseada em LOC

Segundo Roger S. Pressman (2006, p.526), uma pesquisa em dados históricos mostra que a profundidade organizacional média para sistemas desses tipo é de 620 LOC/PM. Com base no valor bruto da mão-de-obra de 8.000 dólares

por mês, o custo por linha de código é de aproximadamente 13 dólares. Com base na estimativa de LOC e nos dados históricos de profundidade, o custo total estimado para o projeto é de 431.000 dólares e o esforço estimado de 54 pessoas-mês (as estimativas são arredondadas para o próximo milhar de dólares e pessoa-mês. Mais precisão é desnecessária e irreal, dada a limitação da precisão de estimativa).

5.4 Um exemplo de estimativa baseada em FP

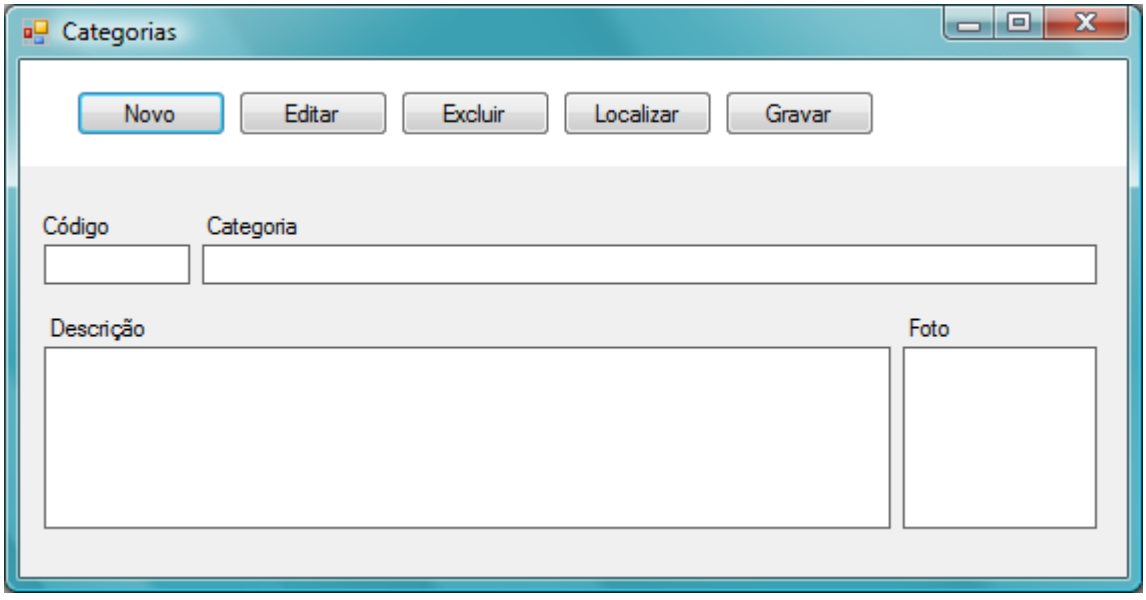
A produtividade organizacional média para sistema desse tipo é de 6,5 FP/PM. Baseado em um valor bruto salarial de 8.000 dólares por mês, o custo por FP é de aproximadamente 1.230 dólares. Baseado na estimativa LOC e nos dados históricos de profundidade, o custo total estimado do projeto é de 461.000 dólares e o esforço estimado de 58 pessoas-mês (PRESSMAN, 2006, p.528).

6 ESTUDO DE CASO

Durante um mês foram medidos e coletados dados de partes de um projeto. Foram contados os pontos de função baseados no ponto de vista do usuário. O tempo de realização dos pontos de função foi registrado em horas. Tudo isso com o objetivo de criar uma base de dados histórica, que sirva como apoio para as estimativas de quanto tempo irá levar para desenvolver determinada funcionalidade do sistema.

Logo abaixo estão as telas que serviram como base para a contagem dos pontos de função e tempo de realização para o desenvolvimento das funcionalidades.

Cadastro de categorias de produtos



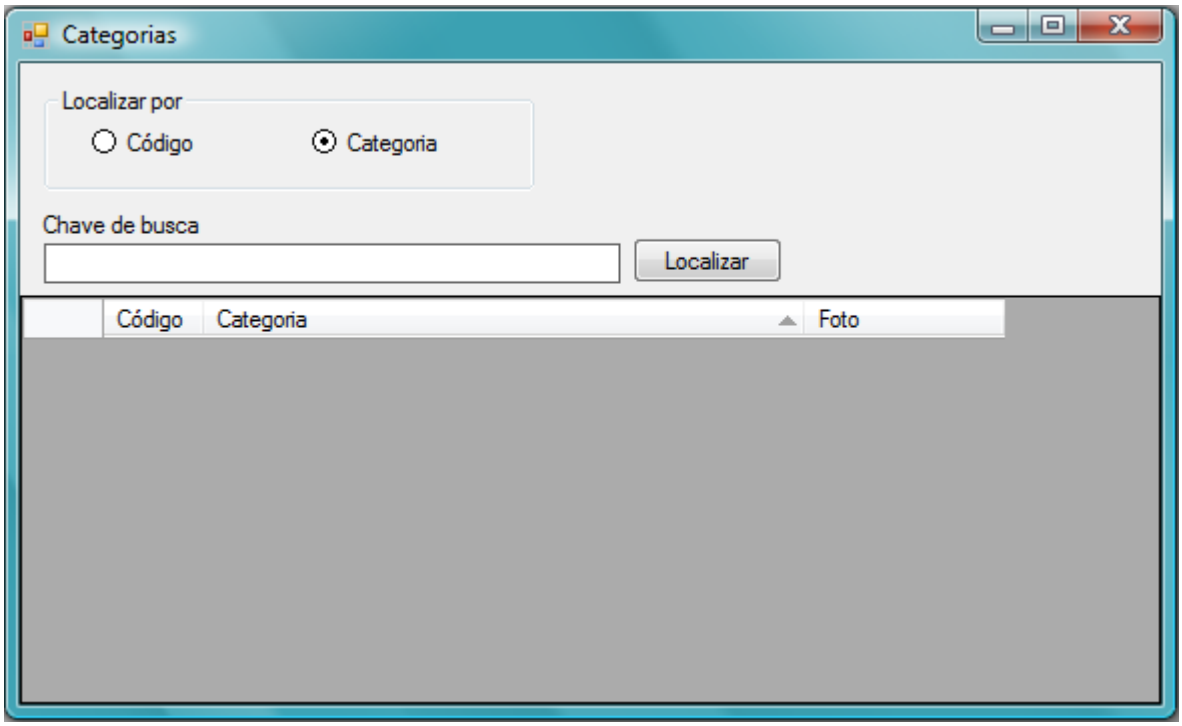
Contagem

Processo Elementar ou Grupo de Dados	*Tipo*	*TD*	*AR/TR*	*Complx.*	*PF*	*rt*
Categoria	ALI	4	1	Baixa	7	2

Inclusão de Categoria	EE	4	1	Baixa	3	1
Alteração de Categoria	EE	4	1	Baixa	3	1
Exclusão de Categoria	EE	4	1	Baixa	3	1
Localização de Categoria	CE	4	1	Baixa	3	1

rt = tempo realizado

Consulta de Categorias

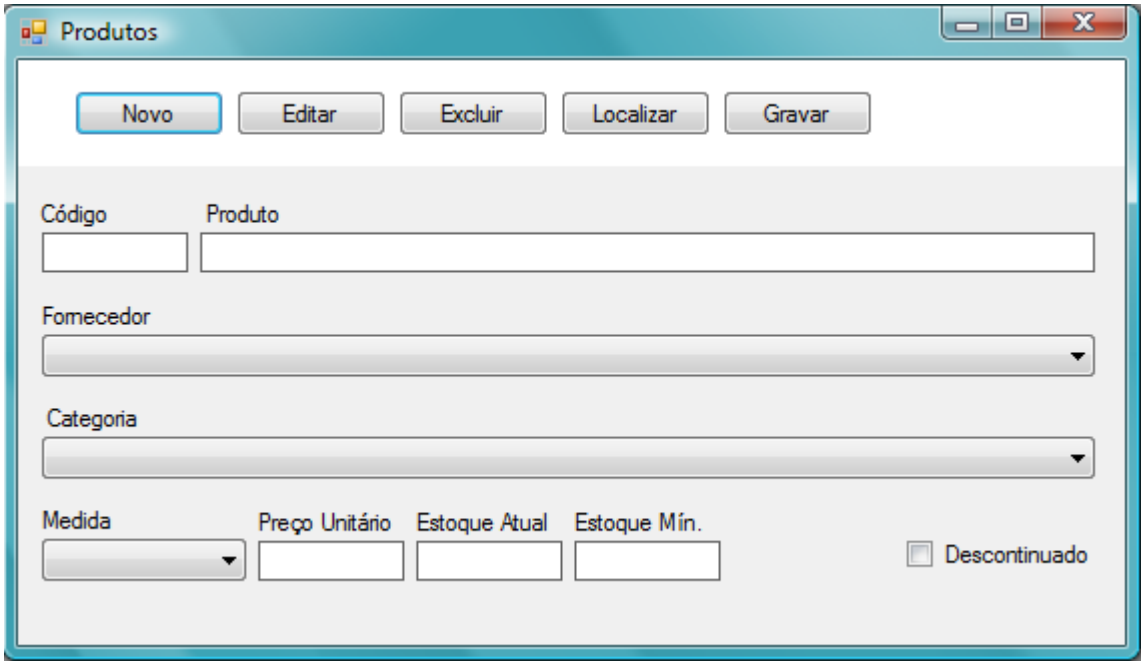


Contagem

Processo Elementar ou Grupo de Dados	*Tipo*	*TD*	*AR/TR*	*Complx.*	*PF*	*rt*
Localização de Categoria	CE	5	1	Baixa	4	1

rt = tempo realizado

Cadastro de Produtos

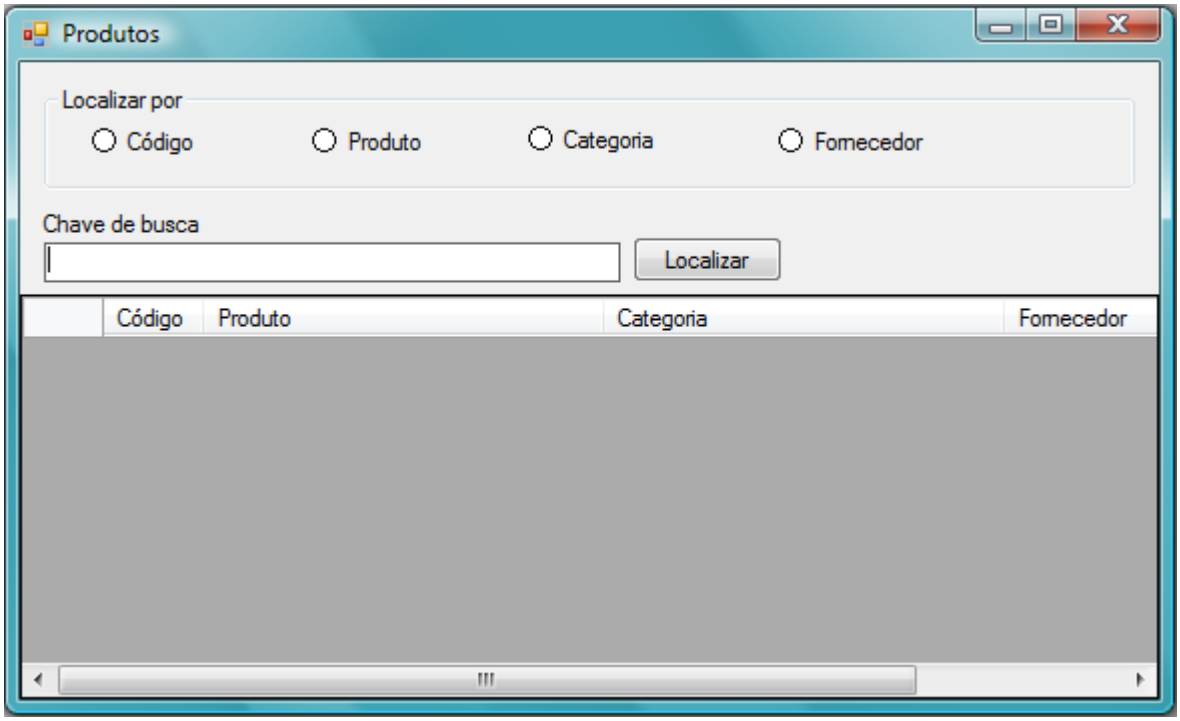


Contagem

Processo Elementar ou Grupo de Dados	*Tipo*	*TD*	*AR/TR*	*Complx.*	*PF*	*rt*
Produto	ALI	9	1	Baixa	7	2
Inclusão de Produto	EE	9	1	Baixa	3	1
Alteração de Produto	EE	9	1	Baixa	3	1
Exclusão de Produto	EE	9	1	Baixa	3	1
Localização de Produto	CE	9	3	Média	4	1

rt = tempo realizado

Consulta de Produtos

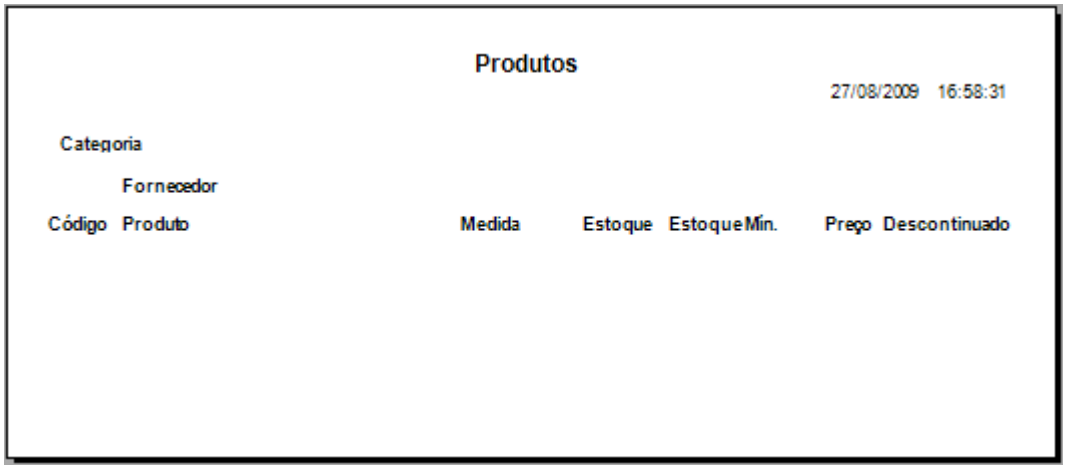


Contagem

Processo Elementar ou Grupo de Dados	*Tipo*	*TD*	*AR/TR*	*Complx.*	*PF*	*rt*
Localização de Produtos	CE	6	3	Média	5	2

rt = tempo realizado

Relatório de Produtos



Contagem

Processo Elementar ou Grupo de Dados	*Tipo*	*TD*	*AR/TR*	*Complx.*	*PF*	*rt*

Emissão de Relatório de Produtos	SE	9	3	Média	5	2

rt = tempo realizado

Os dados coletados acima foram reunidos para formar a base de dados histórica abaixo.

Base de Dados Histórica

Processo Elementar ou Grupo de Dados	*Tipo*	*TD*	*AR/TR*	*Complx.*	*PF*	*rt*	***∆t***
Categoria	ALI	4	1	Baixa	7	2	**2**
Produto	ALI	9	1	Baixa	7	2	**2**
Inclusão de Categoria	EE	4	1	Baixa	3	1	**1**
Alteração de Categoria	EE	4	1	Baixa	3	1	**1**
Exclusão de Categoria	EE	4	1	Baixa	3	1	**1**
Inclusão de Produto	EE	9	1	Baixa	3	1	**1**
Alteração de Produto	EE	9	1	Baixa	3	1	**1**
Exclusão de Produto	EE	9	1	Baixa	3	1	**1**
Localização de Categoria	CE	4	1	Baixa	3	1	**1**
Localização de Categoria	CE	5	1	Baixa	4	1	**1**
Localização de Produto	CE	9	3	Média	4	1	**1**
Localização de Produtos	CE	6	3	Média	5	2	**2**
Emissão de Relatório de Produtos	SE	9	3	Média	5	2	**2**

rt = tempo realizado; ∆t = tempo estimativa calibrada

A tabela Pontos de Função X Tempo, abaixo, demonstra a relação entre a média de pontos por função e a média de tempo em horas, de acordo com a complexidade e tipo de cada um. Estes dados foram calculados a partir da tabela de dados históricos acima.

Pontos de Função X Tempo

Complexidade	*Tipo*	*∆PF*	*∆t*
Baixa	ALI	7	2
	CE	3,5	1
	EE	3	1
Média	CE	4,5	1,5
	SE	5	2

∆PF = média de pontos por função; ∆t = tempo estimativa calibrada

Agora, para demonstrar a estimativa de tempo utilizando a tabela de Pontos de Função X Tempo, observe a coluna ∆***t*** na tabela Base de Dados Histórica. Estes valores foram calculados a partir da relação entre pontos por função e tempo, levando em conta a complexidade e o tipo de cada processo elementar ou grupo de dados.

Se selecionado como exemplo, o processo elementar ou grupo de dados Localização de Produtos, de complexidade média e tipo CE, 5 pontos de função foram equivalentes a 2 horas para sua realização. Na tabela Pontos de Função X Tempo, para complexidade média e tipo CE, a média de pontos de função ∆*PF* é de 4,5 e a média de tempo ∆*t* é de 1,5. Sendo assim, utilizando a regra de equivalência, 5 pontos de função serão equivalentes a 2 horas estimadas, isto arredondado o resultado.

7 CONCLUSÃO

Medição resulta em mudança cultural. Coletar dados, calcular e analisar métricas são três passos que devem ser implementados para iniciar um programa de métricas. Em geral, uma abordagem orientada a metas ajuda a organização a focalizar as métricas adequadas para o negocio. Criando ma referência para métricas, um banco de dados que contêm medições de produto e processo, engenheiros de software e seus gerentes podem ter melhor compreensão do trabalho e do produto.

O planejador do projeto de software deve estimar três coisas antes de iniciar um projeto: quanto tempo ele vai levar, quanto esforço exigirá e quantas pessoas estarão envolvidas. Além disso, o planejador deve estimular os recursos (hardware e software) necessários e o risco envolvido.

A declaração de escopo ajuda o planejador a desenvolver estimativas utilizando uma ou mais técnicas situadas em duas categorias amplas: decomposição e modelagem empíricas. Técnicas de decomposição exigem um delineamento das principais funções do software, seguindo por estimativas de (1) o número de LOC, ou (2) valores selecionados do domínio da informação, ou (3) o número de casos de uso, ou (4) número de pessoas-mês necessário para implementar cada função, ou (5) o número de pessoas-mês necessário para cada atividade de engenharia de software. Técnicas empíricas usam expressões derivadas de resultados práticos para tempo e esforço a fim de prever essas quantidades relativamente ao projeto. Ferramentas automatizadas podem ser usadas para implementar um modelo empírico especifico.

Estimativas de projeto precisas usam, geralmente, pelo menos duas dessas três técnicas. Pela comparação e conciliação das estimativas derivadas utilizando diferentes técnicas, é mais provável que o planejador consiga derivar uma estimativa precisa. A estimativa de projetos de software nunca será uma ciência exata, mas uma combinação de bons dados históricos e técnicas sistemáticas podem melhorar a precisão da estimativa.

REFERÊNCIAS BIBLIOGRÁFICAS

CALDIERA, G et al. Definition and experimental evaluation of Function Points for object-oriented systems. In: International Symposium on Software Metrics, 5. **Proceedings.** IEEE. March 20 – 21, 1998, Bethesda, Maryland. p. 167 – 179.

FENTON, N., PFLEEGER, S. **Software Metrics A Rigorous & Practical Approach**. Boston: PWS Publishing Company, 1997.

GARMUS,D. HERRON, **Function Points Analysis – Measurement Practices for Successful Software Projects**. Estados Unidos: Addison Wesley, 2001.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** São Paulo: Atlas, 2002.

HAZAN, C. Medição da Qualidade e Produtividade em Software. In: Qualidade e Produtividade em Software, 4ª edição, K.C. Weber, A.R.C. Rocha, C.J. Nascimento (organizadores), Makron Books, 2001, p. 25 – 41.

IFPUG. International Function Point Users Group. **Function Point Counting Practices Manual: Release 4.2.** Ohio: IFPUG. 2004. 1 v.

INTERNATIONAL FUNCTION POINT USER GROUP. **Análise por pontos de função.** [S.1.]:IFPUG, 1991. (Baseado na Release 3.4 do Manual de Práticas de Contagem do IFPUG.

MCPHEE, C. **SENG 621:** Software process management: software size estimation. University of Calgary, 1999. 11p.

PRESSMAN, Roger S. **Engenharia de software.** 6 ed., 2006.

PUTMAM, L.; MYERS, W. **Measures for Excellence, Yourdon Press,** 1992.

SILVA I. J. de M.. **Delphi 5 – Análise de Pontos por Função.** Rio de Janeiro, Book Express, 2000.

VAZQUEZ, C. E.; SIMÕES G. S.; ALBERT R. M. **Análise de Pontos por Função.** São Paulo, Érica, 2003.

Consulta na Internet:

AZEVEDO, Douglas José Peixoto de. Disponível na Internet: <http://www.SoftwareMetrics.com> 29 de março de 2003.

BIBLIOGRAFIA

ANDRADE, E. L. P. **Pontos de Casos de Uso e Pontos de Função na gestão de estimativas de tamanho de projetos de software orientados a objetos.** Brasília: UCB. 2004. Tese (Mestrado).

AGUIAR, M. A **Produtividade dos Projetos de Desenvolvimento.** *Developers* Magazine, fev. 2003.

ALESSANDRO, A. **Planejamento e Acompanhamento de Projetos – Os Principais Problemas Típicos.** Revista Developers, n. 81, ano 7, p. 15-16, maio, 2003.

FARIAS, L. D. Planejamento de riscos em ambientes de desenvolvimento de software orientados a organização. Rio de Janeiro: UFRJ/COPPE. 2002. Tese (Mestrado).

GUSTAFSON D. A.. **Engenharia de Software.** Porto Alegre, Bookman, 2003.

HAUFE, M. I. **Produtividade no desenvolvimento de software.** In: Semana Acadêmica do Programa de Pós Graduação em Computação. RS, 1999.

PAULA FILHO, W. P. **Engenharia de Software – fundamentos, métodos e padrões.** Rio de Janeiro, LTC, 2003.

SOMMERVILLE, I. **Engenharia de Software.** Addison Wesley, 2003.