Melhoria Contínua de Estimativa de Esforço para o Desenvolvimento de Software

Ricardo Ajax Dias Kosloski[1], Káthia Marçal de Oliveira[2]

[1,2] Universidade Católica de Brasília UCB SGAN 916 – Brasília, DF

rikoslos@brturbo.com.br, kathia@ucb.br

Abstract. *Look for the continuous improvement of the effort estimate precision can drive the organization to improve its capacity do carry out its commitments, delivering its software products on time and, therefore, bring competitive advantage. We argue that defining a framework of characteristics that impact on software project productivity can improve comparison between finished projects and the new ones that need an effort estimate. This article presents an approach to effort estimate with continuous improvement of the estimates. It is also presented an application of this approach.*

Resumo. *Buscar a melhoria contínua da precisão das estimativas de esforço pode direcionar a organização a melhorar a sua capacidade de cumprir com os seus compromissos, entregando seus produtos dentro dos prazos previstos e, portanto, obter vantagens competitivas. Acreditamos que definir adequadamente as características que causam impactos nas produtividades de projetos de software pode melhorar comparações entre projetos realizados e novos projetos a terem seus esforços estimados. Este artigo apresenta uma abordagem de melhoria para as estimativas de esforço, e sua aplicação numa organização, baseada na idéia de Fábricas de Experiências.*

1. Introdução

Produzir software com alta qualidade, mínimo custo e alta produtividade é crítico para o desempenho empresarial e tem exigido cada vez mais a atenção dos gerentes [Simões 1999]. A precisão das estimativas de esforço é importante, pois, se por um lado, valores superestimados elevam os prazos e os custos dos projetos, prejudicando a competitividade das empresas; valores subestimados causam agendas mal dimensionadas e possibilitando perdas ou prejuízos financeiros [Agarwal 2001]

Uma forma de estimativa usada atualmente relaciona o tamanho do software ao esforço para produzi-lo através da produtividade, isso é: esforço = produtividade x tamanho [Fenton e Pleeger 1997]. O tamanho do software pode ser identificado pelo seu conteúdo funcional, mensurado por métricas, tais como: Análise de Pontos de Função - APF [IFPUG 2005, NESMA 2005] e *Full Funtion Points* [COSMIC 2003]. Contudo, definir a produtividade adequada para as estimativas de um novo projeto de desenvolvimento de software ainda é um desafio para muitas empresas.

Valores de produtividade podem ser encontrados em bases históricas internacionais [ISBSG 2004, Maxwell 2001]. Contudo, cada organização deve ter seus próprios registros históricos de produtividades, refletindo suas atuações junto aos seus clientes

[Farley 2002]. A semelhança verificada em projetos de software, pelas suas características, pode levar ao uso de valores de produtividades mais contextualizados em novas estimativas e proporcionar a melhoria das suas precisões.

A melhoria contínua da qualidade de processos tem sido buscada ao longo do tempo [Johnson 2002], para gerar melhores produtos e buscar a satisfação dos clientes [Bruckhaus 1996]. Nesse contexto surge o conceito de Fábricas de Experiências, a fim de institucionalizar a aprendizagem coletiva da organização para a melhoria contínua e obtenção de vantagens competitivas [Basili *et al* 1994].

Este trabalho apresenta uma abordagem de melhoria contínua para definição de estimativas, que se baseia na caracterização adequada da produtividade e na contínua avaliação das estimativas a partir da coleta de métricas e do registro de lições aprendidas, usando a abordagem da Fábrica de Experiência. Nas próximas seções são apresentados conceitos sobre estimativas de esforço e produtividade (seção 2) e sobre melhoria contínua baseada em Fábrica de Experiências. Na seção 4 e 5 é detalha e aplicada a proposta deste trabalho. A seção 6 apresenta as conclusões e trabalhos futuros.

2. Estimativas de esforço e produtividade

Pelo PMBOK (2004), esforço é a quantidade de trabalho necessário para completar uma atividade ou elemento de um projeto, podendo ser expresso como um total de tempo (horas, dias, etc), gasto por um grupo de pessoas na realização de suas atividades.

As relações simples de estimativas são simplificações de modelos paramétricos, baseadas em dados locais e equações aritméticas, ao invés de modelos matemáticos complexos [McGarry *et al* 2001 p.93]. Nessa relação, a precisão das estimativas depende da precisão da medição de tamanho e da produtividade, em representar o contexto próprio do desenvolvimento. Por usar explicitamente a produtividade, essa forma de estimativa foi escolhida como foco deste estudo. Dessa forma, a produtividade pode ser definida como a divisão do esforço de desenvolvimento do software pelo seu respectivo tamanho funcional [Garmus e Herron 2000] que, por sua vez, pode ser mensurado por métricas, tais como: APF, NESMA (2005), COSMIC.

A APF foi escolhida como métrica de tamanho funcional, pois alem de ser extensamente utilizada em estudos comparativos [Jones 1994], ela: pode ter seus elementos identificados desde cedo pelos requisitos do sistema [Dekkers e Aguiar 2000]; tem sido utilizada como referência de tamanho em cálculos de valores de produtividades [Maxwell 2001] e permite a construção de bases históricas [Farley 2002]. Sobre o tamanho, analisamos apenas fatores que causam impactos no erro de estimativas e não características desses erros causados pela aplicação de métodos diferentes de medição funcional.

A produtividade pode ser obtida de bases históricas internacionais. O David Consulting Group [DCG 2005] consolida valores de taxas de entrega por plataforma de operacionais (grande porte, cliente/servidor, WEB), enquanto o Software Productivity Research [SPR 2001] os apresenta classificados por níveis de linguagens de programação. Taxa de entrega é definida como a quantidade de produto (PF) entregue na unidade de tempo (usualmente em meses)– (PF/mês) [Garmus e Herron, 2001]. O *Institute of Software Benchmarking Standards Group* [ISBSG 2004] apresenta uma base de dados com mais de 2000 registros contendo as produtividades mensuradas a partir da

realização de projetos de software, além de classificá-las com 51 características dos projetos, o que constitui uma taxonomia própria do instituto.

3. Melhoria contínua usando Fábricas de experiências

BASILI *et al* (1994) definiram a fábrica de experiências com base no empacotamento e reuso de produtos e experiências advindos da realização de ciclos de vida de processos de software. A fábrica de experiências é uma organização lógica e física apoiada por dois grandes conceitos: o relativo à evolução – paradigma de melhoria contínua da qualidade (*Quality Improvement Paradigm* – QIP) e a medição e controle, estabelecido pela abordagem *Goal Question Metric* – GQM.

O QIP enfatiza a melhoria contínua pela aprendizagem a partir das experiências obtidas, em nível de projeto e em nível da organização e construídas a partir da experimentação e da aplicação de medições. São definidas as etapas: ***(i)*** **caracterizar** o projeto e seu ambiente com respeito aos modelos e métricas; ***(ii)*** **definir objetivos** quantificáveis para evidenciar as melhorias; ***(iii)*** **selecionar o processo** apropriado para a melhoria, ***(iv)*** **Executar** os processos, construindo os produtos, coletando e validando os dados; ***(v)*** **analisar** os dados para avaliar as práticas atuais, determinando problemas e registrando recomendações para os projetos futuros; e, ***(vi)*** **empacotar** as experiências.

No QIP o GQM é um mecanismo para definir e avaliar o conjunto de objetivos operacionais por meio de medições [Basili *et al* 1994]. No GQM, para medir de forma objetiva a organização deve especificar quais os objetivos a serem alcançados com as medições estabelecidas, que direcionam a elaboração de questões para, depois de refinadas, resultarem em métricas, num modelo "*top-dow*" de definição. A aplicação das métricas deve responder as questões e os objetivos de medição identificados. Os resultados dos dados coletados possibilitarão um modelo "*bottom-up*" de interpretação relacionado com os objetivos estabelecidos [Basili *et al* 1994].

4. Melhoria contínua das estimativas de esforço no desenvolvimento de software

Nessa seção é apresentada a definição de uma fábrica de experiências para as estimativas de esforço de forma a obter melhoria contínua [Kosloski e Oliveira, 2005a, 2005b].

4.1. Fase de caracterização

Para caracterizar os projetos foi elaborado um *framework* de características através da investigação em estudos já realizados sobre o assunto (seção 4.1.1); de entrevistas com especialistas em estimativas de esforço (seção 4.1.2) e de análises em bases históricas internacionais (seção 4.1.3).

4.1.1 Investigações sobre o assunto

Observou-se que a produtividade é essencial para a definição de esforço e que os autores consultados apontam diversos fatores, com diferentes pontos de vista sobre suas influências[1]. Alguns autores mostram que o tamanho da equipe, assim como a

[1] Referências conforme descrito na tabela 2

experiência dos seus integrantes é importante para a definição da produtividade. Outros defendem que restrições de agendas impostas aos projetos e a complexidade do software causam impactos na produtividade. O tamanho do software é considerado relevante nas estimativas por Agarwal (2001). O uso de ferramentas CASE, de metodologias [Potok 1995] e o nível de reutilização pode melhorar a produtividade. Neste trabalho o foco sobre reuso centrou-se na economia de esforço existente ao reutilizar partes do software já construídas, que é diferente do enfoque da APF ao considerar o esforço adicional para desenvolver partes do software para ser usado em outras aplicações [IFPUG, 2005]. As linguagens de programação são importantes na definição da produtividade, direcionando os dados do SPR e sendo uma característica apresentada pelo ISBSG. Os níveis de completude e detalhamento de requisitos foram apontados como fatores importantes, mas suas medições não são triviais [DAVIS *et al* 1993]. Não se considerou este assunto no trabalho pelas dificuldades de medições e sua amplitude de estudo.

4.1.2 Entrevistas com Especialistas

Foram entrevistados 2 especialistas, com experiência de mais de 7 anos em estimativas e mais de 15 anos em analise de sistemas em desenvolvimento de software, que foram advertidos sobre o foco das entrevistas centrar-se nas relações simples de estimativas. Foram feitas duas 2 seções de 4 horas com cada um, iniciadas através das seguintes questões: (i) o que influencia na precisão das estimativas de esforço de projetos de desenvolvimento de software? (ii) como cada um dos fatores influencia nesta precisão?

Inicialmente foi identificado que a precisão das estimativas de esforço depende tanto da precisão das estimativas de tamanho, quanto da precisão das produtividades em representar o contexto real de cada desenvolvimento (figura 1). Nesta análise foi usado o diagrama de espinha de peixe por representar uma estrutura hierárquica relacionando as causas e as incidências complexas do problema e facilitar a sua visualização e discussão [Ishikawa 1982]. Cada fator identificado era, por sua vez, analisado sobre seus próprios fatores de impacto, e assim por diante, para todos os ramos do diagrama.

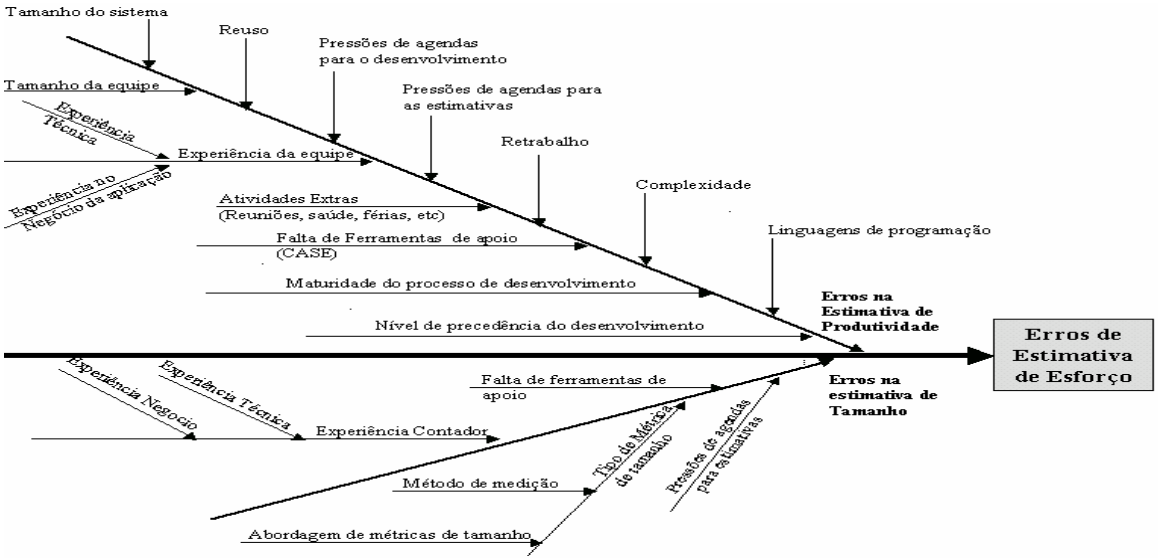

Figura 1: Diagrama de Ishikawa - precisão das estimativas de esforço

Para o tamanho foram identificados aspectos que podem causar impactos nas estimativas e, desta forma, podem ser razões de erros. Com relação às produtividades, as discussões mostraram que existem problemas de erros de estimativas de esforço devido

à influência de diferentes características de projetos de software, que precisam ser conhecidas para direcionar a escolha de valores em novas estimativas. Além disso, algumas características já descobertas tiveram seus pontos de vista refinados. Dentre elas: experiência da equipe que, por opinião dos especialistas, foi subdividida em experiência técnica e nos negócios tratados pela aplicação. Ao final, foi realizada uma sessão geral de 4 horas, para consolidar resultados.

4.1.3 Análise de Bases Históricas Existentes

Um problema ocorre quando os dados são apresentados como taxas de entrega em PF/mês, que não devem ser confundidos com a produtividade em h/PF [Maxwell 2001]. Como o esforço não faz parte da equação de definição de taxas de entrega, devem ser realizadas conversões para obtenção da produtividade em h/PF, onde é usado o número médio de horas trabalhadas mensais. Em ambientes distintos de trabalho, essas horas podem ser diferentes e conduzir a erros nos cálculos de conversão. O DCG e o SPR não informam esse número e são susceptíveis a tais erros, mas não o ISBSG porque as produtividades são calculadas diretamente em horas por pontos de função.

O nível de detalhamento das características cadastradas é importante para a comparação de projetos semelhantes [Maxwell 2001]. O DCG, somente permite comparações entre plataformas operacionais. O SPR produz melhores resultados, apresentando apresentar valores por linguagens de programação. No entanto, somente esta característica não é suficiente para descrever o comportamento das produtividades de projetos de software [Ptunam 2003]. O ISBSG apresenta entre suas 51 características somente a produtividade realizada em cada projeto, sem detalhes sobre a precisão de suas estimativas iniciais, dificultando o estudo do que influencia nas suas precisões.

Neste trabalho optou-se pelo uso do ISBSG por que: vários fatores de impacto estudados estão presentes nas suas características; os seus resultados são apresentados em termos de produtividades (h/PF), não sendo sujeitos às imprecisões de conversões. Além disso, o ISBSG, permite comparações com o SPR e o DCG pelas características: linguagem de programação e plataforma de desenvolvimento, respectivamente. A analise das 51 características do ISBSG, verificou se as mesmas influenciavam ou não a produtividade. Foi observado que:

(i) com base na literatura e na opinião dos especialistas, algumas características não causavam impactos diretamente na produtividade e foram desconsideradas no framework pelos seguintes motivos: não causar impactos diretos na estimativa por serem relativas à qualidade do produto final (*defeitos entregues* e *total de defeitos entregues*). Não terem sido referenciadas pelos estudos já realizados e opiniões de especialistas (*número de usuários envolvidos*, *número de usuários concorrentes* e *data de implementação*). Serem muito genéricas ou com pouca definição (*tipo de linguagem de programação* - 1ª, 2ª, 3ª. ou 4ª. geração)[2], substituída por *linguagem primária de programação* e também *uso de sistemas gerenciados de bancos de dados*, que possui

[2] A agregação de linguagens por geração dificulta avaliações comparativas, o que se constitui numa das principais críticas de bases históricas com dados muito abrangentes tais como o DCG. Além disso, quais seriam os critérios de classificação das linguagens "por geração"? O ISBSG não traça qualquer tipo de comentário sobre como tem sido feita esta classificação

apenas valores do tipo Sim/Não não tendo muito interesse. Terem sido consideradas como ultrapassadas (*abordagens de tabelas de apoio*), importante antigamente por falta de definição da APF sobre tabelas do tipo código/descrição. Algumas não estavam no escopo deste trabalho (*dados de manutenções evolutivas)*, pois considerou-se somente o desenvolvimento de novos softwares e *número de linhas de código*, porque decidiu-se pelo uso da APF.

(ii) outras são redundantes e poderiam ser aprimoradas: *pontos de função não ajustados* e *razão entre eles e os pontos de função ajustados*, pois podem ser derivadas a partir do *tamanho em pontos de função* e do *valor do fator de ajuste. Tipo de organização* e *área de negócios da organização* tem valores similares e optou-se pelo uso da segunda. *Esforço sem divisão de fases e esforço total sumarizado* são idênticas [ISBSG 2004] e optou-se pelo uso da segunda e, finalmente, não foi usada a característica *arquitetura* porque ela tem quase os mesmos dados de *plataforma de desenvolvimento* e *linguagem de programação.*

Outras podiam ser generalizadas: u*so de pacotes customizados* e *níveis de customização mostram* se existiu no projeto o uso de pacotes customizáveis de software, além do quanto de customização foi realizada. Elas foram avaliadas pela característica mais genérica *reuso*, descrita no próximo grupo. *Efetividade do uso do tempo* substituiu a *razão entre o tempo produtivo e não produtivo do projeto*.

(iii) um grupo de características tem alguma influência na produtividade, devendo ser utilizadas na caracterização de projetos de desenvolvimento de software. As que não impactam na produtividade, mas servem para análises de semelhanças (*método de registro de esforço*; *escopo do projeto* e *tempo gasto pelo projeto*). As que causam impactos na produtividade e servem para análises de semelhança (*abordagem de métrica de tamanho*; *métrica de tamanho funcional*; *tamanho máximo da equipe*; *tipo de desenvolvimento*; *plataforma de desenvolvimento*; *linguagem primária de programação*; *técnicas utilizadas no desenvolvimento*; *área de negócios da aplicação* e *tipo de aplicação*). As usadas na avaliação da precisão do processo de estimativas (*tamanho do sistema - PF*; *valor do fator de ajuste da contagem*; *Esforço de trabalho sumarizado - h*; *Esforço de trabalho subdividido por disciplinas - h* e *Produtividade realizada - h/PF)*.

Neste grupo também foram consideradas as características que causam impactos na produtividade, servem para análises de semelhança e foram modificadas ou otimizadas a partir das investigações ou das opiniões dos especialistas: Nível de utilização de ferramentas CASE, avaliada pela escala de FENTON e PFLEEGER (1997), substituindo *ferramentas de apoio de baixo, médio e alto nível*. Paradigma de desenvolvimento, substituindo *metodologia utilizada* (com somente os valores sim / não) e *modo de aquisição da metodologia* (classificadas como: “adquiridas” ou “desenvolvidas internamente”). Categoria de pontos de função, indicando a quantidade de funções (ALI, AIE, EE, CE e SE) existentes na contagem de pontos de função, que foi modificada para, através da relação percentual de elementos de complexidades diferentes (baixas, médias e altas), avaliarem a aplicação sob a visão de *complexidade funcional*. Nível da ferramenta de apoio à contagem de pontos de função, substituindo *tecnologia utilizada para apoiar a contagem*, onde foi usada a escala de classificação de ferramentas de apoio do IFPUG (2005) na sua definição.

A consolidação de fatores que causam impactos na produtividade é apresentada na tabela 2, agrupados por: **(C1)–Projeto de software**, com 4 fatores relativos ao próprio software; **(C2)-Desenvolvimento do software**, com 15 fatores relativos ao processo de desenvolvimento; **(C3)-Tamanho do software**, com 4 fatores sobre às medições ou estimativas de tamanho; **(C4)–Esforço**, com 4 fatores relacionados ao esforço e **(C5)–Experiência da equipe**, com 6 fatores sobre as habilidades dos integrantes da equipe..

Tabela 2. Características de projetos de software

(C1)–Projeto de software	
Características	**Descrições e valores possíveis**
C1.1–Grau de precedência [Boehm 2000]	Avaliado por escala de [Boehm 2000]: Muito baixo, alto e extra alto.
C1.2- Tipo de aplicação [Lim 1994]	Sistemas: transacionais de produção (STP), informações gerenciais (SIG), datawarehouse (DW), etc
C1.3-Complexidade do software [Boehm 2000]	Complexidade funcional, como definida pela [IFPUG 2000] em funcionalidades:simples, médias ou complexas.
C1.4-Tipo de área de negócios [ISBSG 2004]	Contabilidade, área bancária, jurídica, saúde, etc.
(C2)-Desenvolvimento do software	
C2.1-Tipo de desenvolvimento [ISBSG 2004]	Desenvolvimento de novos softwares, manutenções evolutivas, manutenções adaptativas.
C2.2-Plataforma de desenvolvimento [ISBSG 2004]	PC, Grande porte ou Mistas.
C2.3-Paradigma de desenvolvimento [ISBSG 2004]	Estruturado, orientado a objetos.
C2.4-Linguagem primária de programação [ISBSG 2004,SPR 2001]	A linguagem principal usada na construção do software
C2.5-Técnicas de desenvolvimento[ISBSG 2004]	**1**-Modelagem de dados; **2**-Prototipação; **3**-Gerenciamento de projetos; **4**-Métricas; **5**-Testes; **6**-JAD; **7**- Gerência de requisitos.
C2.6-Nível de utilização de ferramentas CASE [Fenton e Pfleeger 1997]	Como definido por [Fenton e Pfleeger 1997]: **0**-nenhuma ferramenta utilizada; **1**-ferramentas utilizadas somente como auxilio para menos de 20% da documentação; **2**-ferramentas utilizadas para documentar ao menos 50% do projeto de alto nível; **3**- ferramentas utilizadas para documentar ao menos 50% do projeto de alto nível e detalhado; **4**-Ferramentas utilizadas para projeto e geração automática de código em pelo menos; **5**- Ferramentas utilizadas para projeto e geração automática de código em pelo menos
C2.7-Nível de reuso [Boehm 2000, Lim 1994]	0 – nenhuma parte do software pronta para ser reutilizada 1 – menos de 20%do software pronto para ser reutilizado 2 – entre 20% e 50% do software pronto para ser reutilizado 3 – entre 50% e 80% do software pronto para ser reutilizado 4 – mais do que 80% do software pronto para ser reutilizado
C2.8-Presões de agenda para o desenvol. [Agarwal 2001]	As restrições impostas pelo cliente no desenvolvimento do software
C2.9-pressões de agendas p/ as estimativas [Agarwal 2001]	O tempo disponível para as estimativas de esforço
C2.10-Nível de maturidade do processo de desenvolvimento [Boehm 2000]	Como definido pelo CMMI: níveis 1,2,3,4 ou 5, para a organização no desenvolvimento dos projetos de software.
C2.11-Tamanho máximo da equipe [Morasca e Giuliano 2002]	O tamanho máximo da equipe durante o desenvolvimento.
C2.12-Taxa de variação do tamanho da equipe [Hamid 1996]	A relação entre tamanhos máximos e mínimos da equipe.
C2.13-Método de registro de dados [ISBSG 2004]	Conforme definições do ISBSG [ISBSG 2004]: **Método A:** Horas registradas diariamente conforme suas atividades; **Método B:** Horas derivadas a partir de registros que indicam a associação de pessoas ao projeto; **Método C:** Somente o tempo produtivo despendido por cada pessoa no

	projeto.
C2.14-Escopo do projeto [ISBSG 2004]	As fases: planejamento, projeto, especificação, construção, testes e implementação, considerados na estimativa de esforço.
C2.15-Prazo do projeto [ISBSG 2004];	O tempo em meses para o desenvolvimento do projeto de software, estimado no início e coletado ao final do projeto.
(C3)-Tamanho do software	
C3.1-Abordagem de métrica de tamanho [IFPUG 2005, NESMA 2003]	FPA-IFPUG ou NESMA
C3.2-Métrica de tamanho utilizada [IFPUG 2005]	IFPUG-3.0, IFPUG-4.0, ou superiores.
C3.3-Tamanho do software [Agarwal 2001,IFPUG 2005, ISBSG 2004]	O valor obtido pela aplicação da APF no início e no fim do desenvolvimento do software
C3.4-Valor do fator de ajuste da APF [IFPUG 2005,ISBSG 2004];	Identificados pela APF/NESMA no início e no fim do desenvolvimento do software
C3.5-Nível da ferramenta de apoio à contagem de pontos de função	Ferramentas podem auxiliar cálculos, evitar erros e melhorar a precisão das contagens de PF. Avaliadas conforme escala de níveis de ferramentas de apoio [IFPUG 2005]. **Nível 1:** O usuário determina manualmente como os elementos são contados e o software age somente como um repositório de dados de contagens; **Nível 2:** O software faz perguntas ao usuário, cujas respostas causam decisões sobre os elementos contados; **Nível 3:** O software realiza automaticamente a contagem de pontos de função usando múltiplas fontes de informação. A interação com o usuário é mínima.
(C4)–Esforço	
C4.1-Esforço total apurado [Agarwal 2001]	Esforço total do desenvolvimento estimado no início e coletado ao final do desenvolvimento do software
C4.2-Esforço p/atividade no ciclo desenvolvimento ISBSG 2004,Maxwell 2001]	Esforço por atividades do ciclo de desenvolvimento (planejamento, projeto, especificação, construção, testes e implementação) estimadas no início e coletadas ao final do desenvolvimento do software.
C4.3-Eficiência do uso do tempo do projeto [Farley 2002];	Percentual do tempo do projeto gasto com: problemas de saúde, férias, treinamentos, reuniões e ausências em geral.
C4.4-Retrabalho [Johnson 1996]	Esforço total registrado pela equipe devido a retrabalho de tarefas
(C5)–Experiência da equipe [BOHEM 2000]	
C5.1-Experiência na medição de tamanho- técnica [Agarwal 2001]; **C5.2**- Experiência na medição de tamanho-área de negócios [Agarwal 2001]; **C5.3**- Experiência no desenvolvimento de software- técnica [Morasca e Giuliano 2002]; **C5.4**- Experiência no desenvolvimento de software- área de negócios [Morasca e Giuliano 2002]; **C5.5**-Experiência em requisitos –técnica [Dekkers e Aguiar 2000]; **C5.6**-Experiência em Requisitos – área de negócios [Dekkers e Aguiar 2000]	
O nível de experiência é obtido pela média da avaliação sobre os integrantes da equipe, usando a escala de Fenton e Pfleeger (1997): **0**-Nenhuma experiência; **1**-Familiaridade (através de cursos ou livros), mas sem experiência prática; **2**-Experiência prática em um projeto (ou com até 20 horas de trabalho); **3**-Experiência prática em vários projetos (ou entre 21 e 100 horas de trabalho); **4** - Totalmente experiente.	

4.2. Definição dos objetivos de medição

Para avaliar quantitativamente a precisão das estimativas de esforço e poder verificar continuamente se a aplicação da abordagem leva às melhorias almejadas, foi construído o objetivo de medição descrito na tabela 3.

Tabela 3: Objetivo de medição para a avaliação da precisão das estimativas

Analisar:	As estimativas de esforço
Com o propósito de:	Avaliar
Com relação a:	Precisão das estimativas
Do ponto de vista do:	Gerente de projetos

Foram focados: o erro absoluto, definido como a diferença entre os valores de uma determinada variável (tamanho, esforço ou prazos) considerando suas medições no início e no fim de um projeto e o erro relativo, definido como a razão percentual entre estes dois valores, mensurado também nos mesmos pontos do projeto.

Tabela 4. Questões e métricas para a analise das estimativas de esforço.

Questões	Métricas
1-Qual o erro de estimativa de tamanho?	**M 1.1**-Erro absoluto de estimativa de tamanho **M 1.2**-Erro relativo de estimativa de tamanho
2-Qual o erro de estimativa de esforço?	**M 2.1-**Erro absoluto entre o esforço estimado e realizado **M 2.2**-Erro relativo entre o esforço estimado e realizado **M 2.3–**Produtividade inicial (estimada) **M 2.4–**Produtividade final (realizada) **M 2.5-**Erro absoluto de produtividade **M 2.6-**Erro relativo de produtividade
3-Qual o erro de estimativa de prazos?	**M 3.1-**Erro absoluto entre prazo estimado e realizado **M 3.2-**Erro relativo entre prazo estimado e realizado
4-Quais as restrições de agendas impostas para as estimativas iniciais do projeto?	**M 4.1-**Rapidez da realização da estimativa inicial de tamanho do projeto de desenvolvimento de software **M 4.2-**Percentual de horas adicionais usadas nas estimativas iniciais do projeto de desenvolvimento de software
5-Quais as restrições de agenda impostas para o projeto de desenvolvimento de software?	**M 5.1-**Percentual de horas adicionais com relação ao esforço total do projeto de desenvolvimento de software (horas adicionais conforme descrito em M4.2)

4.3 Definição do processo de estimativas de esforço

Foi elaborado um processo de estimativas de esforço a partir de propostas encontradas na literatura: *Pratical Software Measurement* - PSM [PSM 2002], AGARWAL (2001) e Barcellos (2003). A tabela 5 apresenta o processo resultante desse estudo.

Tabela 5. Processo de estimativas de esforço

Atividade	Descrição
Escolher uma abordagem de estimativa	Avaliar o nível de detalhamento do desenvolvimento do software para escolher entre os métodos: estimado (NESMA) ou detalhado (IFPUG) de estimativa de tamanho a ser usado no início do projeto.
Planejar as estimativas	Planejar os passos da estimativa, estabelecendo compromissos, agendas, prazos de realização, necessidades documentais, responsabilidades e avaliando o esforço dos profissionais envolvidos.
Aprovar o plano estabelecido	Obter a aceitação e o comprometimento dos gerentes para o plano estabelecido.
Obter a documentação do sistema	Obter a documentação necessária do sistema para a estimativa
Executar a estimativa (ou medição) do tamanho do software.	Usar a APF (ou NESMA) para medir (ou estimar) o tamanho do software.
Executar a estimativa de esforço	Selecionar as características do *framework* para procurar por projetos similares, a fim de escolher um valor de produtividade e calcular o esforço estimado.
Aprovar as estimativas	Apresentar os resultados aos gerentes obtendo as suas aprovações e registrando dados na base histórica da organização para futuras comparações

4.4 Execução, Análise e Empacotamento

Na execução do processo de estimativas de esforço para cada novo projeto de software deverão ser feitas análises de semelhanças, baseadas nas características do *framework* proposto, para obter um conjunto de projetos análogos e suas produtividades. Dessa

análise será escolhida pelo gestor das estimativas uma produtividade, a ser usada na estimativa do novo software. Cada novo projeto será acompanhado com coletas de dados, direcionadas pelas medições estabelecidas, gerando novas analises com resultados que devem ser registrados na base histórica. Executar continuamente este processo acrescentará novos insumos, que servirão para avaliar as práticas atuais, determinar problemas, gravar achados e fazer recomendações para projetos futuros (seção 5).

Dessa forma,s e faz necessário uma base de experiência robusta que possua projetos de diferente natureza. Para forma a base de experiência escolheu-se abordagem "*Dust to Pearls* [Basili *et al* 2002], no qual experiências brutas (*Dust*), obtidas na realização das estimativas são rapidamente disponibilizadas como lições aprendidas cadastradas na fábrica de experiências após análises preliminares, gerando (*mini-perals*) de experiências elaboradas. Com o tempo as lições aprendidas são melhoradas acrescentando-se novas experiências pelas execuções consecutivas do ciclo QIP, transformando-se em experiências generalizadas e refinadas (*Pearls*).

5. Aplicação Prática

Esta abordagem foi aplicada em uma grande empresa de desenvolvimento de software no Brasil que utiliza desde o ano 2000 métricas de tamanho nas suas estimativas de esforço, buscando melhorar o planejamento de projetos de software. Atualmente a organização obtem dados do SPR e do ISBSG e o tamanho do software é medido pela APF [IFPUG 2005] ou NESMA (2003). Tais medições resultam no tamanho em pontos de função, mas a obtida pela NESMA (contagem estimada) é mais simplificada do que a do IFPUG (contagem detalhada) por ser baseada em valores médios de complexidade das funcionalidades.

Foram seguidos os passos: ***(i)*** estender a atual ferramenta da organização para registrar as 34 características definidas no *framework*; ***(ii)*** popular a base de dados com projetos previamente realizados para análises iniciais sobre fatores de impacto nas produtividades; ***(iii)*** simular algumas estimativas de esforço por similaridade considerando os projetos previamente cadastrados e ***(iv)*** realizar estimativas de esforço para novos projetos de software.

No passo ***(i)***, evoluiu-se a ferramenta usada na organização desde 2002, que já suportava contagens de pontos de função com a criação das novas facilidades: cadastrar características, para armazenar o *framework* estabelecido; aplicar as características aos projetos cadastrados, para a coleta de resultados das métricas; registrar as lições aprendidas, a partir das execuções e avaliações realizadas e disponibilizar a lista de projetos caracterizados, onde o usuário escolhe que características deverão ser utilizadas na consulta de semelhança, assim como fornecer seus respectivos parâmetros de seleção.

Rus *et al* (2003) relatam a dificuldade em obter as primeiras experiências e suas lições aprendidas, para motivar o seu uso e sugerem popular a base de dados com informações extraídas da literatura a partir de experiências prévias já existentes. Para popular a base de dados (passo ***ii***), foram analisadas as documentações de 30 projetos de desenvolvimento, realizados nos últimos 5 anos pela organização, e suas estimativas. Foi difícil encontrar projetos documentados com informações suficientes sobre as 34

características definidas. Porém, 7 projetos forneceram definições para 29 características.

Não foi possível obter C2.8 e C2.9, por falta de registros completos das horas extras usadas nas atividades do projeto e nas estimativas. Algumas práticas de transformar horas extras em folgas, resultaram deficiências nos registros de esforços realizados, gerando incertezas sobre seus valores, o que prejudicou também a medição de C4.3. Como existiram deficiências nos registros de esforço por disciplina, a mensuração de C4.2 também não foi possível e a característica C4.4 não foi avaliada porque o conceito de retrabalho[3] não estava estabelecido na organização conforme a literatura pesquisada. Não foram obtidos dados suficientes de outros projetos nas suas documentações ou por entrevistas com gerentes, pois vários deles não trabalhavam mais na organização. A tabela 6 apresenta os resultados da caracterização e a tabela 7 os resultados das métricas.

Com tais informações foi possível iniciar o ciclo QIP para novos projetos com análises para obtenção de lições aprendidas. Foram encontrados erros relativos consideráveis de precisão de estimativas de tamanho entre 8,7% (S6) e 126 % (S5), com valor médio de 59,8 %. A contagem estimada pode ocasionar imprecisões nos resultados, mas estas magnitudes não se justificam segundo estudos comparativos do NESMA (2005).

A APF baseia-se em requisitos funcionais do software para determinar o seu tamanho [IFPUG 2005]. A experiência dos analistas nos negócios da aplicação é também um fator importante para obter requisitos detalhados que apoiem contagens mais precisas [Dekkers e Aguiar 2000]. S6, com os maiores experiências obteve o menor erro de tamanho. S1, S2, S4 e S5 obtiveram níveis mais baixos nesse quesito e piores precisões de estimativas de tamanho, indicando uma possível fonte de erros para a variação de escopo funcional dos sistemas considerados.

Sobre a métrica M4.1, indicando as pressões de agendas para as estimativas, a maior velocidade de contagem foi de S1 (186 PF/dia.contador) e S6 a menor (29,8 PF/ dia. contador). Comparando-se estas informações com os erros relativos de tamanho obtém-se o gráfico da figura 2 (dados de S4 indisponíveis para coleta), onde se observa que, exceto para S1, os erros de estimativas de tamanho acompanham as velocidades de contagens. Não foram identificadas correlações claras entre o tamanho e a produtividade, nem entre o tamanho e os erros de estimativas de produtividades. Contudo, tanto os estudos realizados na literatura (seção 4.1.1), quanto às opiniões dos especialistas (seção 4.1.2) mostram que elas causam impactos na produtividade de projetos de desenvolvimento de software, indicando a necessidade de mais estudos de caso para averiguações.

Para completar a avaliação desta abordagem serão necessários alguns anos desde que as estimativas são realizadas no início dos projetos de software que, após realizados, terão suas medições analisadas. Para se ter uma idéia inicial sobre o potencial de melhoria possível de ser obtido, decidiu-se fazer algumas simulações (passo ***iii***) com os 7 projetos caracterizados. A idéia foi verificar dois projetos similares (tabela 8) e definir a

[3] Retrabalho: qualquer atividade de correção, modificação ou melhoria de um artefato de software já considerado como finalizado (PARK *et al*, 1996). Problemas encontrados durante o desenvolvimento de um módulo ou função, ainda não validado e considerado finalizado, não está sendo considerado retrabalho (GOMES, 2001).

estimativa de esforço de um deles com base na produtividade do outro e verificar se ela seria melhor do que a estimativa realizada na prática pela organização. Pelos erros de estimativas, decidiu-se usar o tamanho final do projeto para as simulações.

Os projetos 1 e 5 foram usados como referências na avaliação das estimativas de outros projetos e é possível ver que, mesmo com esta pequena simulação, pode se ter melhorias nas estimativas de esforço (tabela 9).

Tabela 6: Características aplicadas a projetos de desenvolvimento de software

Caract	S1	S2	S3	S4	S5	S6	S7
C 1.1	Muito baixa	Muito baixa	Alta	Muito baixa	Muito baixa	Extra alta	Muito baixa
C 1.2	SIG	SIG	SIG	SIG	SIG	SIG	SIG
C 1.3	S=71 M=12 C = 17	S=67 M=18 C = 15	S=51 M=23 C=26	S=61 M=18 C=21	S=79 M=13 C=8	S=62 M=14 C=24	S=89 M=0 C=11
C 1.4	Banc /Jur	Banc/Fundo Pensões	Banc/Jur	Banc/Adm	Banc/Adm	Banc/Jur	Saúde
C 2.1	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.
C 2.2	Grande Porte	Grande Porte	PC,mista	PC,mista	PC,mista	PC,mista	PC,mista
C 2.3	Estruturada	Estruturada	Oriet.Obj	Oriet.Obj	Oriet.Obj	Estruturada	Oriet.Obj
C 2.4	Natural	Natural	Java	Java	Java	Cold Fusion	Java
C 2.5	1,2	1,2,3,4,5	1,2,3,4,5,7	1,2,3,4,5,6	1,2,3,4,5	1,2,3,4,5,7	1,2,3,4,5
C 2.6	1	2	3	2	2	3	2
C2.7	0	1	1	1	1	2	0
C2.10	1	1	1	1	1	1	1
C 2.11	22	18	14	14	6	14	9
C 2.12	> 250%	80%	63%	55%	100%	50%	80%
C 2.13	C	C	B	B	B	A	B
C2.14	Todas fases	Todas fases	Todas fases	Todas fases	Todas fases	Todas fases	Todas fases
C2.15i	12	12	16	7	5	11	6
C2.15f	27	17	21	11	8	13	9
C 3.1/ C3.2 i	IFPUG4.1	IFPUG4.1	NESMA	NESMA	NESMA	IFPUG4.1	NESMA
C 3.1 / C3.2 f	IFPUG4.1	IFPUG4.1	IFPUG4.1	IFPUG4.1	IFPUG4.1	IFPUG4.1	IFPUG4.1
C.3.3i (PF)	2324	1967	1710	698	277	2008	224
C3.3f (PF)	3023	3446	2421	1124	625	2183	395
C3.4i	1,12	1,12	1,05	1,01	0,94	1,22	0,97
C3.4f	1,12	1,12	1,05	1,01	0,94	1,22	0,97
C3.5	1	1	1	1	1	1	1
C4.1i (horas)	22032	18647	19152	7818	3103	19678	3164
C4.1f (horas)	49010	43091	32617	13903	8106	21705	7194
C5.1	1,0	2,30	2,5	2,0	3,0	2,5	1,7
C5.2	0,5	1,0	1,5	1,0	1,0	3,8	0,3
C5.3	2,4	3,4	2,6	2,25	1,0	3,0	2,0
C5.4	1,6	2,3	1,8	1,5	0,5	3,0	1,0
C5.5	2,5	2,10	2,3	2,6	2,5	2,7	1,3
C5.6	0,75	1,30	2,7	1,2	1,0	3,7	0,0

Legenda: S: Simples;M: Média;C: Complexa;Banc: Bancária;Jur: Jurídica; Adm: Administrativa; i: Inicial; f: Final; SIG: Sistema de informações gerenciais; PF: ponto de função

Em algumas aplicações o erro ao aplicar a abordagem é negativo, indicando que o esforço estimado teria sido maior do que realmente foi gasto. Porém, considerando apenas os valores absolutos, as diferenças entre valores estimados e realizados teriam sido menores. Assim sendo, os erros de tamanho são fatores preponderantes para as precisões das estimativas de esforço, podendo fazer com que a aplicação da abordagem resulte em valores excessivos para as estimativas de esforço. Mesmo assim ainda foram verificadas melhorias.

Tabela 7: Resultados de medições

Métrica	S1	S2	S3	S4	S5	S6	S7
M 1.1 (PF)	699	1479	711	426	348	175	171
M 1.2 (%)	30.1	75	42	61	126	8.7	76
M 2.1 (horas)	26978	24444	13465	6085	5003	2027	4030
M 2.2 (%)	123	131	70	78	161	10.3	127
M 2.3 (1) (h/PF)	9.48 (1*)	9.48 (1*)	11.20 (1*)	11.20 (1*)	11.20 (1*)	9.8 (2*)	14,1 (2*)
M 2.4 (2) (h/PF)	16,2	12,50	13,40	12,40	13,00	9,94	18,2
M 2.5 (h/PF)	6,70	3,02	2,20	1,20	1,80	0,14	5,3
M 2.6 (%)	71	32	20	11	16	1,5	40,5
M3.1 (meses)	15	5	5	4	3	2.3	3
M3.2 (%)	136	42	31	57	60	21	50
M4.1(PF/dia.contador)	186	65,2	57	- X -	92,3	29,8	75

Legenda: PF:ponto de função; 1*: SPR; 2*: ISBSG

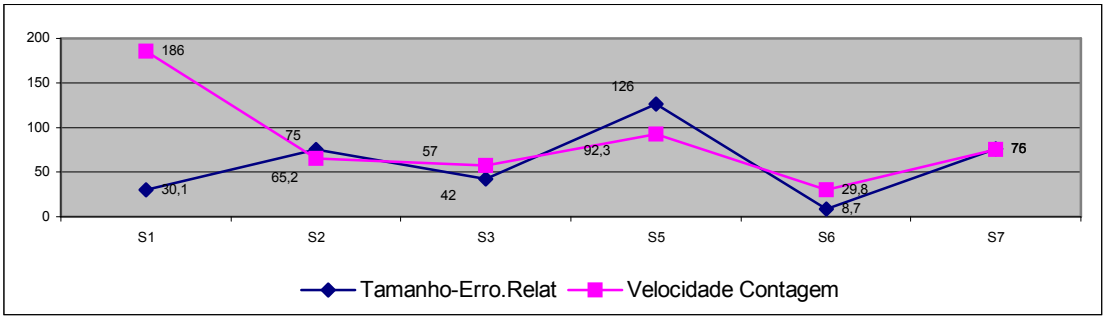

Figura 2: Erros relativos de tamanho e velocidades de contagens

Analisando as semelhanças entre os sistemas considerados (tabela 8) também possibilitou a obtenção de lições aprendidas. Na analogia entre S3 e S5 observa-se que a velocidade das estimativas iniciais de S5 foi 62% maior do que a de S3 (M4.1–tabela 7). Além disso, S5 obteve um dos menores níveis de precedência da amostra, enquanto S3 tem um nível melhor para esta característica (C1.1). Ao se comparar a experiência da equipe, S5, apesar de ter contado com a equipe mais experiente em termos técnicos (medição de tamanho e requisitos) obteve baixos níveis de experiências sobre os negócios tratados pela aplicação. Com os resultados dos passos ***(i)*** a ***(iii)*** as experiências foram cadastradas na base histórica de estimativas de esforço para uso futuro.

O passo (iv), prevendo a realização de estimativas de esforço para novos projetos de software atualmente encontra-se em andamento. A aplicação do *framework* gerou as seguintes recomendações e ações na organização:

- Evolução dos padrões existentes das estruturas analíticas dos projetos (EAP) [PMBOK 2004] atualmente usadas nos cronogramas, visando melhorar as medições;

- O uso do *framework* para identificar a atual precisão do processo de estimativas de esforço, obtendo conhecimentos para comparações futuras.
- A evolução da ferramenta para um sistema corporativo (WEB), disponibilizando suas facilidades ao longo de toda a organização.
- Investimentos em estratégias para melhorar a gestão de requisitos e obter suas métricas associadas.

Tabela 8: Sistemas similares (algumas características avaliadas em magnitudes)

Sistema	Analogias
1 e 2	C1.1; C1.2; C2.1; C2.2; C2.3; C2.4; C2.10; C2.11; C2.13; C2.14; C2.15; C3.3 (diferença no tamanho inicial menor que 20%); C3.4; C3.5; C4.1
3 e 5	C1.2; C2.1; C2.2; C2.3; C2.4; C2.5 (somente uma técnica diferente utilizada); C2.7; C2.10; C2.13; C2.14; C3.1; C3.2; C3.5
3 e 1	C1.2; C1.4; C2.1; C2.10; C2.14; C3.1; C3.2; C3.4; C3.5; C4.1
4 e 5	C1.1; C1.2; C1.4; C2.1; C2.2; C2.3; C2.4; C2.5 (somente uma técnica diferente utilizada); C2.6; C2.7; C2.10; C2.13; C2.14; C3.1; C3.2; C3.4; C3.5

Tabela 9: Simulações comparativas de estimativas de esforço

	Usando SPR			Usando abordagem proposta		Esforço realizado (horas)	Erros de precisão (%)	
Sistema	Tamanho final (PF)	Produtiv. SPR (h/FP)	Esforço (h)	Produtividade Histórica (h/FP)	Esforço (h)		SPR	Abordagem proposta
S2	3446	9.48	32668	De S1 (16,2 h/FP)	55825	43091	31,9	-22,8
S3	2421	11,20	27115	De S1 (16,2 h/PF)	39220	32617	20,3	-16,1
S3	2421	11.20	27115	De S5 (13.0 h/FP)	31473	32617	20,3	3,6
S4	1124	11.20	12589	De S5 (13.0 h/FP)	14612	13903	10,4	-4,8

6. Conclusões e trabalhos futuros

As estimativas de esforço constituem um insumo fundamental para a gerência de projetos de software e a sua melhoria conduz na melhoria da capacidade competitiva da organização para cumprir com seus compromissos de prazos. Este artigo apresentou uma abordagem para a melhoria contínua das estimativas de esforço usando o conceito de Fábricas de Experiências, gerando as contribuições: framework de características, para escolha de produtividades de projetos semelhantes em novas estimativas; uma aplicação da fábrica de experiências, para a melhoria contínua do processo de estimativas e um conjunto de métricas, para avaliar a precisão das estimativas de esforço. Além disso, as atividades realizadas para o estabelecimento da abordagem apresentada condizem com as descritas pelo CMMI – nível 4 (quantitativamente gerenciado), onde são identificados processos críticos, para serem avaliados quantitativamente a fim de identificar oportunidades de melhoria.

Assim sendo, esperamos que, a cada novo projeto, seja melhorada a precisão das estimativas de esforço, através do uso de valores de produtividades mais realísticos, obtidos a partir dos projetos similares, realizados e registrados na base histórica da organização. Como próximos passos, buscamos novas formas de avaliar a similaridade entre projetos com o uso de Raciocínio Baseado em Casos - RBC [Aamodt 1991]. Além disso, estudos futuros devem considerar: a inclusão de outros focos de complexidade; comparações envolvendo métodos diferentes de medição, e seus respectivos impactos na precisão das estimativas e avaliar a proposta para adaptá-la a projetos de manutenção.

Agradecimentos

Este trabalho é apoiado pelo CNPq, uma instituição do governo brasileiro para o desenvolvimento científico e tecnológico.

Referências Bibliográficas

AAMODT, Agnar; PLAZA, Enric; CASE-BASED Reasoning: Foundational Issues, Methodological Variations and Systems Approaches, AI Communications, IOS Press, Vol. 7:1, pp. 39-59, 1991

AGARWAL, Manish, Kumar; YOGESH, S. Mallick; BRARADWAJ, R. M., et all, Estimating Software projects, ACM SIGSOFT, p.60, 2001

BARCELLOS, Monalessa, P., Planejamento de Custos em Ambientes de Desenvolvimento de Software Orientados à Organização, UFRJ, 2003

BASILI, Victor, COSTA, Patrícia, LINDVALL Mikael, MENDONÇA, Manoel, et al, "An Experience Management System for a Software Engineering Organization", IEEE, 2002

BASILI,V.;CALDIERA,Gianluigi;ROMBACH,H.Dieter , The Experience Factory", Encyclopedia of Software Engineering, John Wiley & Sons, 1994, V.1 pp. 476-496

BOEHM, Barry et al,Software Cost Estimation With COCOMO II, Prentice Hall PTR, 2000

BRUCKHAUS et al, 1996, BRUCKHAUS, Tilmann, MADHAVJI, Nazin H., JANSSEN, Ingrid, HENSHAW, John, The impact of Tools on Software Productivity, IEEE,1996

COSMIC, Measurement Manual, The COSMIC Implementation Guide for ISO/IEC 19761:2003, V 2.2, 2003

DAVIS, Alan OVERMYER, Scott; JORDAN, Kathleen; *et al*, Identifying and Measuring Quality in a Software Requirements Specification, IEEE, 1993

DCG, David Consulting Group, retrieved from Internet by the link http://www.davidconsultinggroup.com/ indata.htm in Fev/2005

DEKKERS, Carol; AGUIAR, Mauricio; Using Function Points Analysis (FPA) do Check the Completeness (Fullness) of Functional User Requirements, Quality Plus, 2000

FARLEY,Dick. Making Accurate Estimates, IEEE, 2002

FENTON, N., PFLEEGER, S. Software Metrics A Rigorous & Practical Approach, 2nd. Ed., PWS Publishing Company, 1997.

GARMUS,HERRON, David; HERRON, David. Function Point Analysis – Measurement Practices for Sucessful Software Projects, Addison-Wesley Information Technology Series, 2000

HAMID,Tarek K. Abdel,The Slippery Path to Productivity Improvement,IEEE Software, 1996

IFPUG, CPM – Counting Practices Manual, release 4.2.1; IFPUG; 2005

ISBSG. International Software Benchmarking Standards Group. The Benchmarking, Release 8, ISBSG,; 2004

Ishikawa, Kaoru, Guide to Quality Control, Asian Productivity Organization, Tokyo, 1982

JOHNSON, Donna I.; BRODMAN, Judith G.; Realities and Rewards of Software Process Improvement, IEEE 1996

JONES, C. Software Challenges: Function point: a new way of looking at tools, Computer, Aug 1994. p.66 –67

KOSLOSKI, Ricardo, OLIVEIRA, Káthia M., An Experience Factory to Improve Software Development Effort Estimates, PROFES - 2005a, pp560-573

KOSLOSKI, Ricardo, OLIVEIRA, Káthia M., Melhoria Contínua de Estimativa de Esforço para o Desenvolvimento de Software, SBQS 2005b.

LIM, Wayne C., Effects of Resue on quality, Productivity, and Economics, IEEE, 1994

MAXWELL, Katrina D.; Colleting Data for Comparability: Benchjmarking Software Development Productivity, IEEE Software, Setembro/Outubro 2001

MCGARRY, John, CARD, David, JONES, Cheryl, LAYMAN, Beth, CLARK, Elizabeth, DEAN, Joseph, HALL, Fred, Practical Software Measurement (PSM) – Objective Information or Decision Makers, Addison Wesley, 2002

MORASCA,GIULIANO,Sandro, GIULIANO, Russo. An Empirical Study of Software Productivity, IEEE,2002

NESMA, Estimate Counting, Netherlands Software Metrics Users Association, http://www.nesma.nl/english/index.htm, acessado pela internet em 31/03/2005

POTOK,VOUK, Tom; VOUK, Mladen. Development productivity for comercial SW Using OO Methods; ACM, 1995

RUS,Ioana, LINDVALL, Mikael, SEAMAN, Carolyn, BASILI, Victor, Packaging and Disseminating Lessons Learned from COTS-Based Software Development, IEEE, 2003

SIMÕES, C. Sistemática de Métricas, Qualidade e Produtividade, Developers' Magazine, Brasil, 1999

The PMBOK Guide – 2004 edition, PMI - Project Management Institute, 2004;