Melhoria Contínua de Estimativa de Esforço para o Desenvolvimento de Software

Ricardo Ajax Dias Kosloski , Káthia Marçal de Oliveira

PRPGP–Pró-Reitoria de Pós-Graduação e Pesquisa–Universidade Católica de Brasília
UCB - *Campus* Universitário II - SGAN 916 - Asa Norte. 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 to 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 de software 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.*

1 Introdução

Produzir software de alta qualidade e com o mínimo custo possível, isto é, com alta produtividade, é crítico para o bom desempenho empresarial e tem exigido cada vez mais a atenção dos gerentes [Simões 1999]. Neste sentido, 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 de desenvolvimento; por outro, valores subestimados causam agendas mal dimensionadas e com possibilidades de perdas ou prejuízos financeiros [Agarwal 2001, Hamid 1999].

Uma forma de estimativa bastante empregada atualmente são as relações simples de estimativas, onde o esforço é relacionado ao tamanho do software por meio da produtividade pela equação: *esforço = produtividade x tamanho do software* [Fenton e Pleeger 1997]. A medição de tamanho funcional está bem estabelecida com o uso de diferentes métricas, tais como: Análise de Pontos de Função – APF [IFPUG 2000; NESMA 2003] e Full Funtion Points [COSMIC 2003]. Contudo, definir a produtividade ainda é considerado um desafio para muitas empresas.

Medir objetivamente como as características do desenvolvimento afetam a produtividade é importante para o entendimento e a melhoria das estimativas[Maxwell 2001]. Segundo esta autora, é importante ter certeza de se estar comparando coisas iguais, pois o

desconhecimento das características dos projetos que geraram os valores de produtividades registrados nas bases históricas fragiliza o processo de estimativas, pelo risco de se estar comparando valores obtidos em condições diferenciadas. Jorgensen e Ostvold (2004) defendem a necessidade de identificar fatores que influenciam nos erros de estimativas para melhorar sua precisão. Idealmente, cada organização deve ter os registros de seus dados históricos de produtividade, refletindo suas atuações em desenvolvimento de software junto aos seus clientes [Farley 2002]. A semelhança, verificada pelo registro destas características, quando comparadas com novos desenvolvimentos sendo estimados, poderá levar à utilização de melhores valores de produtividades para cada novo caso de estimativas de esforço.

Este artigo 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. Para isso, é utilizado a abordagem de fábrica de experiência proposta por Basili *et al* (1994). Nas próximas seções são apresentadas breves descrições sobre o uso de produtividade em estimativas de esforço (seção 2) e sobre Fábrica de Experiências (seção 3). Em seguida são apresentadas as definições e uso da Fabrica de Experiências para a melhoria contínua da precisão das estimativas de esforço (seção 4). A seção 5 apresenta uma aplicação desta abordagem em uma grande empresa de desenvolvimento de software. A seção 6 apresenta as conclusões e trabalhos futuros.

2 Usando produtividades nas estimativas de esforço

De forma genérica, esforço pode ser definido como a quantidade de trabalho necessária para completar uma atividade ou outro elemento de um projeto, podendo ser expresso como um total de horas, dias, meses ou semanas, gastas por um grupo de pessoas na realização de suas atividades [PMBOK2000]. Nas relações simples de estimativas, onde esforço e tamanho se relacionam de forma direta e linear [Garmus e Herron 2000]. Desta forma, a produtividade pode ser definida como a divisão da quantidade de trabalho gasto no desenvolvimento do software (esforço em horas) pelo seu respectivo tamanho funcional [Fenton e Pfleeger 1997,Garmus e Herron 2000] que, por sua vez, pode ser mensurado por vários tipos de métricas, tais como: APF, NESMA, COSMIC.

Neste trabalho a Análise de pontos de função (APF) é usada com métrica de tamanho funcional, pois alem de ser extensamente utilizada e consistente em estudos comparativos [Jones 1994], ela: pode ter seus elementos identificados desde cedo a partir dos requisitos do sistema [Dekkers e Aguiar 2000]; é independente da tecnologia adotada na solução [IFPUG 2000], permitindo comparações entre arquiteturas, ferramentas ou processos de software [Agarwal 2001]; tem sido utilizada como referência de medida de tamanho do software em cálculos de valores de produtividades [ISBSG 2004, Maxwell 2001] e permite a constituição de bases históricas de sistemas avaliados [Farley 2002,Maxwell 2001]. Com esta visão, focamos no estudo da produtividade e, com relação ao tamanho, analisamos apenas fatores que causam impactos no seu erro de estimativas e não características dos erros causados pela aplicação de métodos diferentes de medição funcional.

A produtividade não deve ser confundida com a taxa de entrega, uma outra medida de desempenho em projetos de software, que pode ser definida como: o número de pontos de função entregue pela organização de desenvolvimento em uma determinada unidade de tempo (usualmente em meses) [Garmus e Herron 2000]. Para as produtividades, quanto maior for o número (horas por pontos de função - PF) menor é a produtividade e, no caso de se estar trabalhando com taxas de entrega (PF por mês), este raciocínio é invertido.

Atualmente, as organizações têm extraído valores de produtividades de bases históricas internacionais tais como: o *David Consulting Group* [DCG 2005], o *Software Productivity Research* (SPR) [SPR 2001] e do *Institute of Software Benchmarking Standards Group* (ISBSG) [ISBSG 2004]. Tanto o *David Counsulting Group* – DCG, quanto o *Software Productivity Research* - SPR apresentam valores de *taxas de entregas*, em pontos de função por mês. O DCG consolida valores por plataforma de desenvolvimento (grande porte-13 PF/mês, cliente/servidor-17 PF/mês, WEB-25 PF/mês), enquanto o SPR os apresenta classificados por níveis de linguagens de programação. Para cada linguagem de programação o SPR também apresenta o número médio de instruções de código fonte, necessários para implementar cada ponto de função (PF). O *Institute of Software Benchmarking Standards Group* - ISBSG apresenta uma base de dados com mais de 2000 registros contendo as produtividades (em horas por PF), mensuradas a partir da realização de projetos de software. Além disso, também são apresentadas 51 características de projetos, que constituem a taxonomia própria do ISBSG para o registro de valores.

3 Fábrica de experiências

Em 1994, BASILI *et al* (1994) definiram o conceito de 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 a evolução – paradigma de melhoria contínua da qualidade (*Quality Improvement Paradigm* – QIP) e o relativo a medição e controle, como estabelecido pela abordagem *Goal Question Metric* – GQM.

O QIP enfatiza a melhoria contínua por meio da aprendizagem a partir das experiências obtidas, tanto em nível do projeto, quanto em nível da organização, construídas a partir da experimentação e da aplicação de medições. O QIP considera seis etapas: (i) **caracterizar** o projeto e seu ambiente com respeito aos modelos e métricas; **(ii) definir objetivos** quantificáveis a fim de 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.

Uma forma apropriada e sem ambigüidades para a caracterização do ambiente é um pré-requisito para a correta aplicação do paradigma [Basili *et al* 1994]. Isto requer que o projeto de software seja classificado com respeito a um conjunto de características de forma a poder isolar projetos similares em novas estimativas. Além disso, a caracterização provê um contexto de reuso de experiência e produtos, seleção de processos, avaliações, comparações e predições.

A abordagem GQM é o mecanismo do QIP para definir e avaliar o conjunto de objetivos operacionais por meio de medições [Basili *et al* 1994]. Segundo a abordagem GQM, para medir de forma objetiva, deve especificar quais os objetivos a serem alcançados com as medições estabelecidas [Basili *et al* 1994]. Tais objetivos direcionam a elaboração de questões que, depois de refinadas, resultam em métricas, cuja aplicação responderá as questões estabelecidas e, conseqüentemente, os objetivos de medição identificados.

4 Melhoria contínua das estimativas de esforço no desenvolvimento de software

Com a necessidade de proporcionar melhores estimativas para cada novo projeto de desenvolvimento de software, e inspirados pela idéias de melhoria contínua do desenvolvimento embasada anteriormente, decidimos definir e construir uma fábrica de experiências para as estimativas de esforço. Nas seções seguintes são apresentados como

definimos as fases de caracterização, definição de objetivos e seleção de processos. Os outros passos são executados para cada novo projeto.

4.1 Fase de caracterização

Nesta fase foi necessário definir como seriam caracterizados os projetos de desenvolvimento que necessitassem de estimativas de esforço. Neste sentido, elaboramos um framework de características a partir de: investigações em estudos já realizados sobre o assunto e seus resultados (seção 4.1.1);entrevistas com especialistas em estimativas de esforço (seção 4.1.2) e análises em bases históricas internacionais (seção 4.1.3).

4.1.1 Estudos em pesquisas relevantes sobre o assunto

Várias abordagens e pesquisas têm considerado diferentes fatores importantes na definição da produtividade:

- Restrições de prazos: para as estimativas de esforço [Agarwal 2001,Putnam 2003] e no desenvolvimento do software [Agarwal 2001,Hamid 1996];
- Tamanho da equipe de desenvolvimento [Hamid 1996, Morasca e Giuliano 2002];
- A complexidade do software a ser construído, que pode ser de vários tipos [Fenton e Pfleeger 1997]:complexidades computacionais, algorítmicas, estruturais, cognitivas e funcionais, conforme APF e Pontos de caso de uso – PCU, [IFPUG 2000,Kirsten 2001];
- Experiência da equipe, tanto em nível das técnicas utilizadas, quanto nos negócios tratados pelo software afeta a produtividade [Boehm 2000,Morasca e Giuliano 2002];
- O reuso afeta a produtividade [Lim 1994], causando o aumento de produtividade, tanto para no paradigma orientado a objetos, quanto no estruturado [Potok 1995];
- O uso de metodologias pode reduzir o tempo do projeto por otimizar tarefas e proporcionar melhores entendimentos sobre o software [Potok 1995];
- O uso de ferramentas pode ter impacto positivo ou negativo nas produtividades, dependendo de fatores adicionais tais como: treinamentos, configurações especiais no ambiente de desenvolvimento, etc [Boehm 2000];
- Técnicas utilizadas nos desenvolvimentos, conforme mencionado por RUBIN (1993);
- O tamanho como fator de impacto na produtividade, é considerado por [Agarwal 2001, Boehm 2000];

Pode-se observar que a produtividade é essencial para a definição de esforço em projetos de software e que os autores consultados apontam diversos fatores com diferentes pontos de vista sobre suas influências e comportamentos.

4.1.2 Entrevistas com Especialistas

Completando os estudos da seção anterior, foram feitas entrevistas com 3 especialistas, reconhecidos como experientes em estimativas, com mais de 15 anos de experiência em analise de sistemas em desenvolvimento de software e também com mais de 7 anos na área de estimativas de esforço. Os especialistas foram advertidos sobre o foco das entrevistas ser relativo à equação: *Esforço (h) = produtividade (h/PF) x tamanho (PF)*.

Cada especialista foi entrevistado em duas seções, de 4 horas cada, iniciadas por meio das seguintes questões provocativas: ***(i)*** quais os fatores que influenciam na precisão das estimativas de esforço de projeto de desenvolvimento de software?, ***(ii)*** como cada um destes fatores influencia nesta precisão?

Duas grandes raízes de fatores para os erros de estimativas foram identificadas: a precisão das estimativas de esforço depende tanto da precisão das estimativas de tamanho, quanto da precisão dos valores de produtividades no contexto real de cada desenvolvimento.

A partir de então as análises foram feitas sobre os fatores de impacto no tamanho e na produtividade.

Nesta análise foi desenhado um diagrama de espinha de peixe [Ishikawa 1982] para melhor visualizar o problema. Este diagrama representa uma estrutura hierárquica relacionando as causas e as incidências complexas de um problema para facilitar a sua visualização e discussão. Cada fator identificado era, por sua vez, analisado sobre seus próprios fatores de impacto, e assim por diante, para todos os ramos do diagrama. A figura 2 apresenta o diagrama final obtido pelas entrevistas, mostrando as influencias no tamanho e na produtividade. Vários fatores podem influenciar a produtividade, como, por exemplo, experiência da equipe, tanto técnica como gerencial.

As entrevistas mostraram algumas características já descobertas pelos estudos realizados na seção anterior e outras novas, as quais motivaram novas pesquisas. Dentre as novas características ressaltam-se: nível de precedência do desenvolvimento (também avaliado pelo COCOMO II [Boehm 2000]), maturidade do processo de software (também estudados em [Rubin 1993]) e nível de retrabalho (conforme discutido em [Johnson 1996]). Além disso, algumas características já descobertas tiveram novos pontos de vista agregados. Dentre elas ressaltam-se: experiência da equipe [Boehm 2000,Morasca e Giuliano 2002] que, por opinião dos especialistas, foi subdividida em experiência técnica e nos negócios tratados pela aplicação. Os especialistas também concordaram com algumas características provenientes dos estudos da seção anterior, tais como: restrições de agendas para o desenvolvimento do software [Putnam 2003], uso de ferramentas CASE [Boehm 2000] e complexidade da aplicação [Maxwell 2001]. Ao final, foi feita uma sessão de 4 horas, realizada com todos os envolvidos, para consolidar os resultados obtidos.

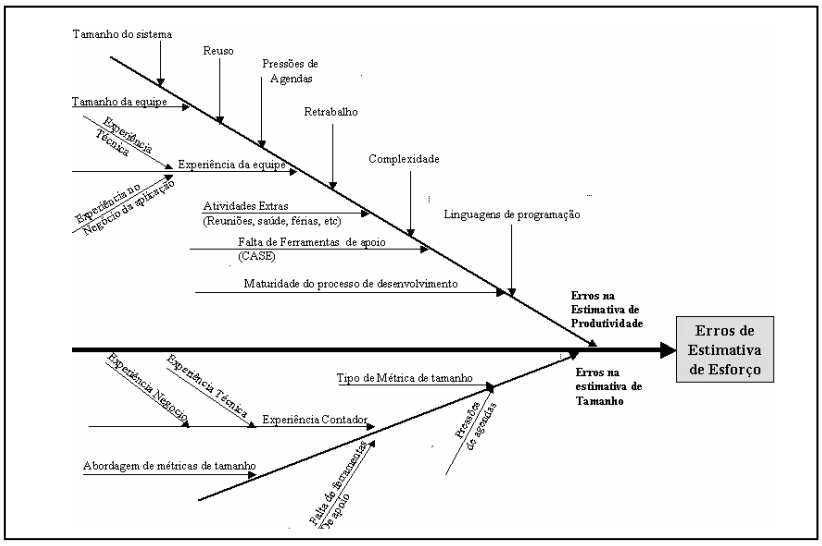

Figura 2: Diagrama de Ishikawa para o problema de precisão das estimativas de esforço

4.1.3 Análise de Bases Históricas Existentes

Como citado na seção 3, diferentes bases históricas existem para a definição de produtividade. No entanto, alguns aspectos importantes devem ser destacados.

Um primeiro problema ocorre quando os dados são apresentados em termos de taxas de entrega [Maxwell 2001], conforme descrito na seção 2. Como o esforço não faz parte da equação de definição de taxas de entrega, para que sejam usadas as relações simples de estimativas, antes elas devem ser convertidas em valores correspondentes de produtividades. Como um exemplo de conversão, suponhamos uma taxa de entrega de 13 PF/mês num

contexto onde o número médio de horas produtivas ao mês seja de 160 horas. Então, a produtividade correspondente será o resultado da operação: 160 (h/mês) / 13 (PF/mês), ou seja, será de 12,3 h/PF.

Existem dois pontos críticos neste raciocínio. Inicialmente o número médio de horas produtivas disponíveis ao mês pode variar em contextos locais de trabalho. O COCOMO II considera 152 horas [RUBIN 1993] (US), enquanto o Reino Unido (UK) trabalha com 120 horas médias mensais. FARLEY (2002) também acrescenta que algumas organizações alocam entre 75 e 80% do tempo das suas equipes como disponível, devido a: reuniões, treinamentos, férias, etc. Tais considerações levam às seguintes conversões: 120 (h) / 13 (PF/mês) resultando numa produtividade correspondente de 9,23 h/PF (diferença de -25%).

Tanto o *David Consulting Group*, quanto o SPR não informam este número e, portanto, se tornam susceptíveis a erros advindos deste cálculo. No ISBSG este problema não aparece, pois as produtividades apresentadas são calculadas diretamente dos registros de tamanho e esforço, advindos da realização de projetos de software. Isto nos fez decidir pelo não uso de bases históricas que usassem taxas de entrega e seguir a linha proposta pelo ISBSG. Outro problema particular do SPR consiste na sua forma de apresentação de valores de taxas de entrega, pois existem linguagens de programação classificadas em níveis intermediários da tabela 1 (C: nível 2,5; C++ e Java: nível 6; Visual Basic 5: nível 11, etc). Como o próprio SPR afirma, o número de níveis distribuídos pelas faixas não é linear deforma que este tipo de apresentação conduz à necessidade de usar interpolações lineares para a correta extração de valores de taxas de entrega. No entanto, por não existirem maiores definições sobre tais não linearidades, este processo torna-se difícil e sujeito à imprecisões. A inconveniência dos dados do SPR contribuiu para a nossa decisão de usar a base histórica do ISBSG.

Finalmente, outra questão importante em bases históricas, é o nível de detalhamento das características de projetos cadastrados, pois quanto mais características estiverem disponíveis melhores serão as condições de análises de semelhanças para novos projetos a serem estimados [Maxwell 2001]. O *David Consulting Group*, somente oferece condições para comparações superficiais entre plataformas operacionais e o SPR permite melhores comparações por apresentar valores por linguagens de programação. No entanto, comparações baseadas somente nesta característica não são suficientes para descrever o comportamento das produtividades de projetos de software [Ptunam 2003].

Por outro lado, o ISBSG mesmo apresentando um conjunto de 51 características para cada projeto cadastrado, apresenta somente as produtividades obtidas de projetos de software realizados sem detalhes sobre a precisão de suas estimativas iniciais. Isto dificulta o estudo sobre o que influencia na precisão de estimativas. Analisamos, as 51 características do ISBSG verificando se as mesmas influenciavam ou não a produtividade. Nessa análise observamos que: (i) algumas características não causavam impactos diretamente na produtividade baseado na literatura e opinião dos especialistas, (ii) outras são redundantes e poderiam ser aprimoradas, e (iii) um grupo de características tem alguma influência quando se fala em produtividade devendo ser utilizadas.

No grupo (i), as características foram desconsideradas para o framework pelos seguintes motivos:

- **não causar impactos diretos na estimativa:** como, por exemplo, *defeitos entregues* e *total de defeitos entregues*, por serem relativas à qualidade do produto final.
- **não terem sido referenciadas pelas investigações em estudos já realizados e pelas opiniões de especialistas:** como, por exemplo, *número de usuários envolvidos*, *número de usuários concorrentes* e *data de implementação*.

- **serem muito genéricas:** como, por exemplo, *tipo de linguagem de programação* (1ª, 2ª, 3ª. ou 4ª. geração), substituída por outra conforme proposto no framework (*linguagem primária de programação*); e *uso de sistemas gerenciados de bancos de dados (DBMS)*, que possui apenas valores do tipo Sim/Não não tendo, portanto, muito interesse.
- **Terem sido consideradas como ultrapassadas:** No passado existia alguma discussão sobre a forma de considerar as tabelas de referências (com somente código e descrição) na APF e, portanto, a característica *abordagens de tabelas de apoio* era importante, pois diferentes abordagens poderiam levar a diferentes resultados para o tamanho do software. Atualmente, em versões mais recentes (a partir da 4.1 [IFPUG 2000]), a APF traz formas mais bem estabelecidas para considerar este elemento de forma que isto não é mais um problema.
- **Finalmente, algumas não estavam no escopo deste trabalho**: como, por exemplo, *dados de manutenções evolutivas*, pois estamos considerando somente o desenvolvimento de novos softwares e *número de linhas de código*, porque decidimos usar a APF para identificar o tamanho do software.

No grupo (ii), as características consideradas como redundantes foram aprimoradas ou otimizadas como se segue:

- **Características otimizadas para simplificar o framework**. Como, por exemplo, *pontos de função não ajustados* e *razão entre eles e os pontos de função ajustados*, não foram usadas, 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 no ISBSG e optamos pelo uso da segunda. *Esforço sem divisão de fases e esforço total sumarizado* são conceitualmente idênticas [ISBSG 2004] e optamos pelo uso da segunda e, finalmente, não usamos a características *arquitetura* porque ela tem quase os mesmos dados cobertos por *plataforma de desenvolvimento* e *linguagem de programação.*
- **Características generalizadas.** Como, por exemplo, u*so de pacotes customizados* e *nível de customização* mostram se no projeto existiu o uso de pacotes customizáveis de software e também o quanto de customização foi realizada em tais pacotes. Elas foram avaliadas pela característica mais genérica *reuso*, conforme descrito no próximo grupo. A razão entre o *tempo produtivo* e *não-produtivo* do projeto foi substituída pela similar *efetividade do uso do tempo*.

As características do grupo (iii) foram usadas conforme suas definições originais e agrupadas conforme descrito a seguir:

- **As que não causam impactos nas produtividades, mas servem para análises de semelhança:** como, por exemplo, *método de registro de esforço*; *escopo do projeto* e *tempo de duração do projeto*.
- **As que causam impactos na produtividade e servem para análises de semelhança:** como, por exemplo, *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:** como, por exemplo, *tamanho do sistema (PF)*; *valor do fator de ajuste da contagem (adimensional)*; *Esforço de trabalho sumarizado (h)*; *Esforço de trabalho subdividido por disciplinas (h)* e *Produtividade realizada (h/PF)*.

Também neste grupo foram consideradas aquelas 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 consultados:

- Nível de utilização de ferramentas CASE: avaliada segundo a escala de FENTON e PFLEEGER (1997), em substituição à classificação do ISBSG (ferramentas de apoio de baixo, médio e alto nível).
- Paradigma de desenvolvimento em substituição às características do ISBSG: *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 [IFPUG 2000]. Apesar de existirem outros focos de observação sobre a complexidade (lógica, algorítmica, cognitiva, etc) [Fenton e Pfleeger 1997], modificamos esta característica para, por meio da relação percentual de elementos de complexidade diferentes (baixas, médias e altas complexidades), avaliar a aplicação sob a visão de complexidade funcional.

Outro fato observado foram as diferenças que podem ser obtidas nas escolhas das produtividades dependendo das características selecionadas. Considerando toda a amostra de eventos Java do ISBSG, a média aritmética geral de produtividades é de 25,69 h/PF. Restringindo-se a amostra a projetos de novos desenvolvimentos este valor reduz-se para 22,82h/PF (variação de -12,6%) e, considerando também, somente projetos de instituições da área bancária este valor diminui para 12,97 h/PF (variação de -49,5%). É interessante comparar estes valores com os obtido do SPR (11 h/PF) e do *David Donsulting Group* onde, para a plataforma WEB, este valor é de 6,4 h/PF.

A consolidação de características gerou os fatores apresentados na tabela 2, agrupados por: **(C1)–Projeto de software**, com 4 fatores sobre o software a ser desenvolvido; **(C2)-Desenvolvimento do software**, com 15 fatores sobre o próprio desenvolvimento do software; **(C3)-Tamanho do software**, com 4 fatores sobre a APF e NESMA; **(C4)–Esforço**, com 4 fatores relacionados ao trabalho necessário para a construção do software e **(C5)–Experiência da equipe**, com 6 fatores relacionadas a aspectos humanos no desenvolvimento.

Table 2. Características de projetos 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, como definido por [Boehm 2000]
C1.2- Tipo de aplicação [Lim 1994]	Sistemas transacionais de produção, Sistemas de informações gerenciais, etc
C1.3-Complexidade do software [Boehm 2000]	Complexidade funcional, como definida pelo método da APF [IFPUG 2000] em funcionalidades simples, médias ou complexas.
C1.4-Tipo de área de negócios [ISBSG 2004]	Contabilidade, área bancária, jurídica, saúde, etc.
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.

Características	**Descrições e valores possíveis**
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-Restrições de agenda para desenvol. [Agarwal 2001]	As restrições impostas pelo cliente no desenvolvimento do software .
C2.9-Restrições de agendas p/ estimativas [Agarwal 2001]	O tempo disponível para as estimativas de esforço
C2.10-Nível de maturidade do processo de desenvolvimento [RUBIN 1993,Boehm 2000]	Como definido pelo CMMI [20]: níveis 1,2,3,4 ou 5, O nível da organização quando do desenvolvimento dos projetos de software.
C2.11-Tamanho máximo da equipe [ISBSG 2004,]	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 do desenvolvimento de software (planejamento, projeto, especificação, construção, testes e implementação) que foram 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.
C3.1-Abordagem de métrica de tamanho [IFPUG 2000, NESMA 2003]	FPA-IFPUG ou NESMA
C3.2-Métrica de tamanho utilizada [IFPUG 2000]	IFPUG-3.0,IFPUG-4.0
C3.3-Tamanho do software [Agarwal 2001,IFPUG 2000, 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 2000,ISBSG 2004];	Identificados pela APF/NESMA no início e no fim do desenvolvimento do software
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

Características	Descrições e valores possíveis
	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
C5.1-Medição de tamanho-experiência técnica [Agarwal 2001]	Média da avaliação sobre os integrantes da equipe, usando a escala de [13]
C5.2- Medição de tamanho-experiência na área de negócios da aplicação [Agarwal 2001];	0-Nenhuma experiência 1-Familiaridade (através de cursos ou livros), mas sem experiência prática.
C5.3-Desenvolvimento de software-Experiência técnica [Agarwal 2001,Morasca e Giuliano 2002]	2-Experiência prática em um projeto (ou com até 20 horas de trabalho)
C5.4- Desenvolvimento de software- na área de negócios da aplicação [Agarwal 2001,Morasca e Giuliano 2002]	3-Experiência prática em vários projetos (ou entre 21 e 100 horas de trabalho) 4 - Totalmente experiente.
C5.5-Requisitos – Experiência técnica [Dekkers e Aguiar 2000]	
C5.6- Requisitos – Experiência na área de negócios da aplicação [Dekkers e Aguiar 2000]	

4.2. Definição dos objetivos de medição

O seguinte objetivo de medição foi definido:

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

Como previamente definido esforço = tamanho x produtividade e, desta forma, as questões e métricas do GQM foram definidas de forma a avaliar estas duas questões (tabela 3). Foram focados os erros absolutos e relativos nas medições para o entendimento dos seus fatores de impacto. O erro absoluto é a diferença entre os valores de uma variável conforme as medidas no início e no fim de um projeto de desenvolvimento de software e o erro relativo é a razão percentual entre estes dois valores.

Tabela 3. Questões e métricas para a analise das estimativas de esforço.

Questões	Métricas
1-Qual o erro de estimative de tamanho?	**M 1.1**-Erro absoluto de estimativa de tamanho **M 1.2**-Erro relativo de estimativa de tamanho
2-Quais foram as restrições de agendas impostas para a contagem de PF inicial do projeto de desenvolvimento de software?	**M 2.1-**Rapidez da realização da contagem (número de pontos de função contados por dia e por cada contador) **M 2.2-**Percentual de horas adicionais usadas na contagem (“horas adicionais” significam àquelas horas além da jornada diária normal de trabalho)
3-Qual foi o erro de estimativa de esforço?	**M 3.1-**Erro absoluto entre o esforço estimado e realizado **M 3.2**-Erro relative entre o esforço estimado e realizado **M 3.3–**Produtividade inicial (estimada) **M 3.4–**Produtividade final (realizada) **M 3.5-**Erro absoluto de produtividade **M 3.6-**Erro relativo de produtividade
4-Qual foi o erro de estimativa de prazos?	**M 4.1-**Erro absoluto entre prazo estimado e realizado **M 4.2-**Erro relativo entre prazo estimado e realizado
5-Quais foram 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 M2.2)

4.3 Definição do processo de estimativas de esforço

Foi escolhido o processo de estimativas de esforço para a aplicação da idéia da fábrica de experiências, almejando a melhoria da sua precisão. Foram encontradas na literatura duas propostas: a do *Pratical Software Measurement* - PSM [PSM 2002] e o processo descrito por AGARWAL (2001). No PSM, o processo consiste nos seguintes passos: selecionar a abordagem de estimativas, mapear analogias, computar e avaliar estimativas. Neste processo notamos que algumas atividades eram bastante genéricas e deveriam ser mais bem detalhadas (por exemplo: a dependência do método de estimativas no passo "computar estimativas"). Além disso, a atividade aplicada ao final da estimativa ("avaliar estimativas"), apesar de não ser parte integrante das estimativas de esforço, mas nos interessavam no contexto deste trabalho. A proposta de AGARWAL também lida com o processo de uma forma genérica, mas adiciona a importância de bases históricas como fonte de informações úteis na calibração das estimativas. Nosso foco foi definir o processo de estimativas a partir do tamanho do software e da produtividade do seu desenvolvimento conforme apresentado na tabela 4.

Tabela 4. 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, isto é, quem será o responsável pela sua realização, além de quando ela será realizada e qual o esforço será requerido nesta atividade.
Aprovar o plano estabelecido	Obter a aceitação e o comprometimento dos gerentes para o plano.
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.

5 Aplicação Prática

Esta abodagem foi aplicada em uma grande empresa no Brasil que trabalha com estimativas por mais de 5 anos e está constantemente buscando melhores maneiras de estimar esforço para seus projetos de software. Atualmente a organização extrai dados das bases históricas do SPR e do ISBSG. O tamanho do software foi medido pela APF [IFPUG 2000] ou NESMA (2003). Estes métodos resultam no tamanho em pontos de função, mas a medida obtida pela NESMA (chamada de contagem estimada) é mais simplificada do que a do IFPUG (conhecida por contagem detalhada) por ser baseada em valores médios de complexidade das funcionalidades identificadas. Na aplicação desta abordagem foram seguidos os seguintes passos:

i) Estender a atual ferramenta da organização para registrar as 33 características definidas no framework;
ii) Popular a base de dados com projetos previamente realizados como ponto de partida das análises 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.

Para o primeiro passo, foi feita uma manutenção evolutiva na ferramenta atualmente em uso na organização e que já suportava as contagens de pontos de função de projetos de

software. O modelo de dados foi entendido para incluir três novas facilidades: uma para caracterizar projetos de software por meio do uso das 33 características definidas; outra para a pesquisa por semelhança na base de dados e uma para apoiar a coleta de resultados das métricas definidas pela abordagem GQM.

Rus *et al* (2003) os autores reportam que a dificuldade com a fabrica de experiências está em possuir as primeiras experiências registradas na base histórica com suas respectivas lições aprendidas, para motivar o seu uso. Eles sugerem popular a base de dados da fábrica de experiências com informações extraídas da literatura e a partir de experiências prévias já existentes. Desta forma, para popular a base de dados com projetos realizados (passo ii), foram analisadas as documentações de 30 projetos de desenvolvimento de software, realizados nos últimos 5 anos pela organização, e suas respectivas estimativas. Infelizmente devido ao tamanho do framework (33 características), foi difícil encontrar projetos documentados com informações suficientes sobre todas elas. No entanto foram encontradas 27 das 33 características com definições em 7 projetos (não sendo possível avaliar: C2.8, C2.9, C4.2, C4.3 e C4.4). Para os outros projetos não conseguimos informações suficientes, nem na documentação disponível, nem por meio de entrevistas com os gerentes, pois vários deles já não estavam mais a serviço da organização.

A tabela 5 apresenta a caracterização destes 7 projetos. A tabela 6 mostra quais foram os projetos considerados como similares e, na tabela 7, encontram-se os resultados das métricas realizadas. Nestes resultados podem ser vistos que existem erros de estimativas maiores do que 100% (S1, S2, S5 e S7) e erros de prazos estimados maiores do que 50% para três sistemas (S1, S4 e S5). Com esta informação foi possível iniciar o ciclo QIP para novos projetos de software. Para completar a avaliação desta abordagem serão necessários alguns anos desde que as estimativas são realizadas no início dos projetos, que devem ter seus desenvolvimentos completados para então serem analisadas as métricas definidas. Desta forma, para se ter uma idéia inicial sobre o potencial de melhoria possível de ser obtido, decidimos fazer algumas simulações (passo iii) com os 7 projetos já caracterizados. A idéia foi verificar dois projetos similares e definir a 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. A tabela 8 mostra os erros absolutos de estimativas com as produtividades obtidas pelo SPR (usadas pela organização em seus projetos) e esta abordagem. 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 (2 projetos entre 30% e dois entre 70% de melhoria).

Atualmente o passo (iv) está em andamento.

A partir da aplicação deste framework foram feitas recomendações que geraram várias ações da organização. Dentre elas ressaltam-se:

- A evolução dos padrões existentes das estruturas analíticas dos projetos (EAP) [PMBOK 2000] atualmente usadas na elaboração de cronogramas, visando melhorar as medições;
- O uso deste framework no diagnóstico sobre a atual precisão do processo de estimativas, para comparações das melhorias almejadas pelo esforço da organização na melhoria da maturidade dos seus processos de desenvolvimento, por meio do modelo CMMI.
- A evolução da ferramenta de apoio às métricas e estimativas para um sistema corporativo (padrão WEB), para melhor disponibilizar as facilidades da base histórica de produtividades ao longo de toda a organização, visando a melhoria contínua do processo de estimativas.

Tabela 5: Características aplicadas a projetos de desenvolvimento de software

Char.	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	MIS	MIS	MIS	MIS	MIS	MIS	MIS
C 1.3	simples=71 média=12 complexa=17	simples =67 média =18 complexa =15	simples =51 média =23 complexa =26	simples =61 média =18 complexa =21	simples =79 média =13 complexa =8	simples =62 média =14 complexa =24	simples =89 média =0 complexa =11
C 1.4	Bancária jurídica	Bancária Fundo de Pensões	Bancária jurídica	Bancária Administrativa	Bancária Administrativa	Bancária jurídica	Saúde
C 2.1	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.	Desenvolv.
C 2.2	PC, mainframe	PC, midirange	PC,midirange	PC,midirange	PC,midirange	PC,midirange	PC,midirange
C 2.3	Estruturada	Estruturada	Orientada objetos	Orientada objetos	Orientada objetos	Estruturada	Orientada objetos
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 as fases	Todas as fases	Todas as fases	Todas as fases	Todas as fases	Todas as fases	Todas as 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	NESMA	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 (FP)	2324	1967	1710	698	277	2008	224
C3.3f (FP)	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
4.1i (hour)	22032	18647	19152	7818	3103	19678	3164
4.1f (hour)	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	0.50	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	3.5	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

Tabela 6: Sistema similares (algumas características avaliadas em termos de "magnitudes")

Sistema	Analogias
1 and 2	C1.1, C2.8, C1.2, C2.1, C2.3, C2.4, C 3.1, C3.2, C2.13, C2.11, C3.3 (diferença no tamanho inicial menor que 20%), C3.4, C1.3
3 and 5	C1.4, C2.1, C2.3, C2.4, C3.1, C3.2, C2.13, C2.5 (somente uma técnica diferente utilizada)
3 and 1	C1.2, C2.1, C2,8, C3,1, C3.2, C3.3, M2.1, C1.3
4 and 5	C2.8, C3.1, C3.2, C3.3, C2.1, C2.3, C2.4, C2.13, C2.5 (somente uma técnica diferente utilizada)

Tabela 7: Resultados de medições

Métrica	S1	S2	S3	S4	S5	S6	S7
M 1.1 (FP)	699	1479	711	426	348	175	171
M 1.2 (%)	30.1	75	42	61	126	8.7	76

Métrica	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**
M 2.1 (FP/day)	186	65,2	57	- X -	92,3	29,8	75
M 3.1 (hours)	26978	24444	13465	6085	5003	2027	4030
M 3.2 (%)	123	131	70	78	161	10.3	127
M 3.3 (1) (hour/FP)	9.48 (SPR)	9.48 (SPR)	11.20 (SPR)	11.20 (SPR)	11.20 (SPR)	9.8 (*)	12.9 (*)
M 3.4 (2) (hour/FP)	16.2	12,50	13.40	12.40	13.00	9.94	18.2
M 3.5 (hour/FP)	6.70	3.02	2.20	1.20	1.20	0.14	5.3
M 3.6 (%)	71	32	20	11	16	1.5	40.5
M 41 (months)	15	5	5	4	3	2.3	3
M 4.2 (%)	136	42	31	57	60	21	50

Tabela 5: Simulações comparativas de estimativas de esforço

		Usando SPR		Usando abordagem proposta			Erros de precisão (%)		
Sistema	Tamanho Inicial (FP)	Produtiv. SPR (h/FP)	Esforço (h)	Produtividade Histórica (h/FP)	Esforço (h)	Esforço realizado (horas)	SPR	Abordagem proposta	Melhoria
2	1967	9.48	18648	From System 1 (16,2 h/FP)	31865	43091	131	35	**73**
3	1710	11.20	19152	From system 5 (13.0 h/FP)	22230	32617	70	46	35
3	1710	11.20	19152	A partir do sistema 1 (16,2 h/FP)	27702	32617	70	18	**74**
4	698	11.20	7818	A partir do sistema 5 (13.0 h/FP)	9074	13903	78	53	32

6 Conclusões e trabalhos futuros

As estimativas de esforço constituem um insumo fundamental para a gerência de projetos de software. Desta forma, melhorar continuamente a precisão das estimativas leva a organização a também melhorar sua capacidade competitiva, cumprindo com seus compromissos e entregando seus produtos de software nos prazos planejados.

Este artigo apresentou uma abordagem para a melhoria contínua das estimativas de esforço por meio do uso do conceito de Fábricas de Experiências. Esperamos que, a cada novo projeto, sejam melhoradas as precisões das estimativas de esforço, por meio do uso de valores de produtividades mais realísticos, obtidos a partir dos projetos similares já realizados e registrados na base histórica da organização. Como um próximo passo, estamos em busca de novas formas de avaliar comparativamente a similaridade entre projetos já realizados e novos projetos a serem estimados com o uso de Raciocínio Baseado em Casos [Aamodt 1991]. Além disso, estudos futuros sobre o assunto devem considerar a inclusão de outros focos de complexidade, além de estudos comparativos envolvendo métodos diferentes de medição, e seus respectivos impactos na precisão das estimativas.

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

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

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

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 4.1.1; IFPUG – International Function Point Users Group; 2000

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, August, 1994. p. 66 – 67

JORGENSEN, Magne, OSTVOLD, Kjetil Molokken, Rehaznos for software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Análisis Method, IEEE, 2004

KIRSTEN, Ribu; Estimation Object-Oriented Software Projects with Use Cases, University of Oslo, Department of Informatics, 2001

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

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

NESMA, Estimate Counting, Netherlands Software Metrics Users Association, 2003

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

PUTNAM, Lawrence H., MAYERS, Ware, Five Core Metrics – The Intelligence Behind Sucessful Software Management, Dorset House Publishing, 2003

RUBIN, Howard A ., Software Process Maturity: Measuring its Impact on Productivity and Quality, IEEE, 1993

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

SPR, "The Programming Language Table", Software Productivity Research, 2001.

The PMBOK Guide – 2000 edition, PMI - Project Management Institute, 2000;

PSM, Pratical Software Measurement, Adison Wesley, 2002

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