Projeto 2.23

Hercules-parte2: Melhoria Contínua de Estimativa de Esforço no Desenvolvimento de Software baseada na Produtividade

Ricardo Kosloski[1], Kathia Marçal de Oliveira[2]

1 Introdução

O planejamento e controle de projetos de desenvolvimento de software, onde as estimativas de esforço são um importante insumo, têm exigido cada vez mais a atenção dos gerentes. Neste sentido, a precisão da estimativa 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; valores subestimados causam agendas mal dimensionadas e com possibilidades de perdas ou prejuízos financeiros [1].

Uma forma de estimativa bastante empregada atualmente consiste em relacionar o esforço de desenvolvimento ao tamanho do software, por meio da produtividade, de forma que esforço = produtividade x tamanho do software. Nesta equação:

- o esforço, dado em horas (h), é entendido como a quantidade de trabalho a ser executada no desenvolvimento do projeto [7];
- o tamanho pode ser definido a partir do seu conteúdo funcional por métodos como análise por pontos de função (APF) [17;18];
- a produtividade pode ser medida pela relação entre o esforço necessário para a execução do projeto de desenvolvimento e o seu tamanho funcional [18].

Valores de produtividades podem ser encontrados em bases históricas internacionais agrupadas segundo suas próprias caracterizações [12;27]. No entanto o ideal é que cada organização tenha o registro dos seus dados históricos de produtividade, refletindo as suas atuações junto aos seus clientes, no desenvolvimento de sistemas [6]. A semelhança, verificada através do registro das características destas atuações, quando comparadas com novos desenvolvimentos sendo estimados, poderá levar à utilização de melhores valores de produtividades para cada novo caso de estimativas de esforço.

Esse projeto apresenta, uma abordagem de melhoria de estimativa que considera essencialmente a caracterização adequada da produtividade, a fim de melhorar continuamente a precisão das estimativas de esforço, em projetos de desenvolvimento de software.

Nas próximas seções serão apresentados: uma breve descrição sobre as estimativas de esforço através do uso de valores de produtividades (seção 2) e abordagens de melhoria contínua (seção 3). Em seguida será apresentada a abordagem de melhoria contínua definida (seção 4). A seção 5 apresenta as conclusões deste trabalho.

2 Uso de valores de produtividades em estimativas de esforço

A produtividade pode ser definida como a divisão da quantidade de trabalho gasto no desenvolvimento do software pelo seu respectivo tamanho [7;5]. Neste caso, o esforço é dado em horas, enquanto o tamanho pode ser dado por diferentes unidades. Estudos comparativos chegaram à conclusão de que a produtividade obtida através da medida de tamanho com a APF é mais consistente e permite melhores condições de comparação [17;18].

O SPR – Software Productivity Reseach [27] apresenta valores médios de produtividades por linguagens de programação. Por outro lado, o ISBSG – Institute of Software Benchmarking Standards Group [12], apesar de apresentar uma amostra de produtividades para mais de 2000 projetos segundo uma taxonomia própria de caracterízação, não contempla algumas das características apontadas como relevantes na bibliografia (pressões de agendas, experiência da equipe, dentre outros). Além disso, são apresentadas somente produtividades realizadas sem maiores informações sobre a precisão das estimativas iniciais dos projetos cadastrado em sua base histórica.

Diferentes fatores impactam na produtividade. Por exemplo, o tamanho e experiência da equipe de desenvolvimento, tanto na plataforma tecnológica, quanto nos negócios a serem tratados pelo software [6;21;4]. Outra classe de fatores dizem respeito às restrições de prazos impostas ao desenvolvimento [23;9;1].

Alguns autores [9;19] afirmam que simplesmente o registro de dados de produtividade não é suficiente para melhorar o processo de estimativas, sendo necessário analisá-los para entender suas influências em projetos e seus contextos produtivos. Assim sendo, uma vez identificadas as variáveis, ou combinações de variáveis que auxiliam no entendimento dos valores registrados na base histórica de produtividades próprias da organização, elas podem basear a realização das analogias necessárias para as estimativas futuras [19].

4 Abordagens de melhoria

Diferentes abordagens de melhoria vem sendo propostas na literatura. A primeira e mais referenciada foi a proposta de Deming [13] que definiu um ciclo denominado PDCA (Plan-Do-Check-Act). O processo de melhoria era

cíclico e se iniciavam com o planejamento e definição de objetivos, depois a realização do que foi planejado, seguido de uma avaliação para ver se foram atingidos os objetivos e a definição de ações corretivas.

Uma outra abordagem foi proposta pelo SEI [20] que definiu um modelo conhecido como IDEAL que começa desde o momento em que busca o patrocínio da organização para a melhoria até a definição de lições aprendidas. Finalmente, um outro modelo bastante focado para software é a proposta de Basili [2] conhecida como QIP – Quality Improvement Paradigm. O QIP retrata antigas preocupações com a melhoria contínua da qualidade de processos, cujas raízes remontam o final da década de 30, quando o ciclo PDCA (Plan-Do-Check-Act) foi discutido inicialmente,.

O QIP [2] é uma abordagem de qualidade com ênfase na melhoria contínua, através da aprendizagem a partir de experiências de processos na organização e podendo ser construída a partir da experimentação e aplicação de medições. O paradigma QIP é baseado no ciclo PDCA e pode ser detalhado em 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; *(vi) empacotar* as experiências em modelos estruturados.

Para realizar a fase de análise é necessário uma forma objetiva de avaliação. Para isso Basili et al [3] propôs a utilização de uma abordagem denominada Goal Question Metrics. Nessa abordagem são definidos objetivos de medições que são refinados em questões, que definem métricas que devem, por sua vez, fornecer as respostas a estas perguntas.

5 Uma Abordagem de Melhoria Contínua de Estimativa de Esforço em Software

Para definir a abordagem de melhoria decidimos utilizar o paradigma QIP apresentado na seção anterior. A idéia de melhoria contínua consiste basicamente em caracterizar adequadamente o projeto a partir de fatores que realmente influenciam na estimativa de esforço. A partir dessa caracterização, definir um processo de estimativa a ser seguido por todos os projetos. Por meio da execução de um projeto e da coleta de medidas pode-se avaliar os objetivos de melhoria e os possíveis problemas na estimativa, registrando lições aprendidas e informações que sirvam como insumos para melhorar a precisão das próximas estimativas da organização. Executar repetidamente este ciclo imprime neste processo a

melhoria contínua da sua qualidade, que no caso está sendo entendida como a sua precisão.

O primeiro passo para definição da abordagem foi, portanto, definir adequadamente como caracterizar os projetos, que objetivos de melhoria são desejados e qual processo a ser seguido. Os três passos restantes seriam executados a cada projeto com dados particulares de cada um.

5.1 Caracterização

Esta atividade consiste na definição dos fatores de impacto que constituirão o *framework* de caracterização dos projetos de desenvolvimento de software e suas produtividades. Para definir esses fatores, foi realizada uma ampla revisão da literatura e uma série de entrevistas com especialistas na área de estimativas. Os fatores definidos foram agrupados em diferentes categorias conforme apresentados na Tabela 1.

Dessa forma, ao início de cada projeto de desenvolvimento, o projeto deve ser caracterizado segundo esses fatores. A partir dessa caracterização buscam-se na base de experiências projetos com características semelhantes e disponibiliza-se as informações para o gerente de projeto de forma a apoiá-lo nas suas estimativas. A busca de projetos semelhantes, nessa primeira versão da abordagem, é realizada através de um simples consulta de projetos com características iguais às selecionadas pelo gerente, isto é, o gerente escolhe quais características ele quer usar na busca por projetos semelhantes (por exemplo, projetos de mesmo paradigma e com mesmo tamanho), para ser então montada, dinamicamente, uma consulta que varre a base e traz projetos com tais características.

Tabela 1. Caracterização de projetos de desenvolvimento de software

Categoria	**Fatores**
Projetos de software	*1.Grau de Precedência* [4] – medida do quão parecido é o novo projeto com relação a outros já desenvolvidos*;*
	2.Tipo de aplicação [15,16] – Transacionais de produção,gerenciais, etc
	3.Complexidade do software [4,19] – complexidade funcional (APF)
	4. *Tipo de área de negócios* [12,19]~~;~~ – Contabilidade, bancária, saúde, etc
Desenvolvimento do software	*5.Tipo de desenvolvimento* - (novo desenvolvimento, manutenção – evolutiva ou adaptativa) [12]
	6.Plataforma desenvolvimento [12] – Grande

	porte, PC, mista. *7.Paradigma desenvolvimento* [12] – Estruturado, Orientado a Objetos. *8.Linguagem primária de programação* [12,27] *9.Técnicas utilizadas no desenvolvimento* [12] - JAD, Modelagem de dados,etc *10.Nível de utilização de ferramentas CASE* [14] – Avaliada pela escala da tabela 4 *11.Nível de reutilização* [4,15] *12.Restrições de tempo para estimativas* [1,9] *13.Restrições de tempo para o desenvolvimento* [6,23] *14.Nível de maturidade do processo de desenvolvimento* [4,25] *15.Tamanho máximo da equipe* [1,4,16] *16.Taxa de variação do tamanho máximo da equipe* [9,12] *17.Método de registro de dados* – Segundo classificação do ISBSG [12] *18.Escopo do projeto* – segundo classificação do ISBSG [12] *19.Prazo do projeto* [12]
Tamanho do software	*20.Tamanho do software* [1,5,8,10,12,16,17,18,21,23] *21.Abordagem de métrica de tamanho* (APF-IFPUG, NESMA) [10, 17,18,22] *22.Métrica de tamanho utilizada* (IFPUG-3.0,IFPUG-4.0) [10,12] *23Fator de ajuste da contagem de pontos de função*[10,12]
Esforço de trabalho	*24Esforço total apurado* [1,9,16,19,21] *25Esforço por atividade de desenvolvimento* (gerência, análise, programação, testes e implantação) [12] *26.Eficiência da utilização do tempo* (Porcentagem de tempo despendido com outros fatores - problemas de saúde, férias e indisponibilidades em geral) [6,26] 27.*Retrabalho* [29]
Experiência da equipe	Experiência nas técnicas necessárias ao desenvolvimento e nos negócios tratados pela aplicação: 28 e 29.medição de tamanho [1,4,17] (técnica e negocial) 30 e 31.estimativas [1,4,17] (técnica e negocial) 32 e 33.levantamento de requisitos [4,5,17,18]

(técnica e negocial)

5.2 Definição de Objetivos

Foi definido o seguinte objetivo medição:

Analisar: as estimativas de esforço
Com o propósito de: Entender
Com respeito a: precisão das estimativas e suas causas
Sob o ponto de vista da: Gerência de projetos
No contexto de*:* produção no esquema de fábricas de software
Considerando que esforço=tamanho X produtividade como destacado anteriormente, definimos questões e métricas que avaliam o tamanho e a produtividade (tabela 2) e respondam a este objetivo.

Tabela 2. Questões e Métricas para a análise das estimativas

Questões	Métricas
Qual o erro na estimativa de tamanho?	Erro Absoluto em PF (relacionando medição inicial e final do projeto de software); Erro Relativo em % (entre as mesmas medições do item anterior)
Qual a pressão de agendas pelos resultados da medição de tamanho	Velocidade de contagem (número de PF contados por dia, por contador de pontos de função – PF); Percentual de horas extras utilizadas na contagem
Qual o erro de estimativa de esforço?	Valores estimados e realizados para: Esforço; Produtividade e Prazos. Erros absolutos (diferença entre valores realizados e estimados); Erros relativos (Relações percentuais entre valores realizados e estimados).
Qual a restrição de tempo do desenvolvimento?	Percentual de horas extras com relação ao total de horas de esforço apurado (realizado)

5.3 Selecionar o Processo

Como a abordagem de melhoria proposta refere-se à melhoria de esforço, decidiu-se que o processo a ser utilizado para ter sua execução acompanhada é o próprio processo de estimativas. Neste contexto duas propostas são interessantes:a do PSM – Personal Software Measurement [24] e do AGARWAL *et al*[1]. Nessas propostas as atividades são muito genéricas requerendo um melhor detalhamento (ex: computar estimativas)

e outras referem-se à atividades realizadas após a estimativa (ex: avaliar estimativas). Dessa forma, decidimos por definir um processo específico tendo como base essas propostas e o que se conhece sobre estimativas de projeto. O processo resultante está apresentado na Tabela 5.

Tabela 5. Processo de estimativas

Atividade	
Escolher abordagem de estimativa	
Planejar a estimativa	
Aprovar cronograma	
Obter documentos do sistema	
Estimar / medir tamanho	
Estimar o esforço	
Aprovar estimativa	

5.4 Aplicação Prática

Esta abordagem está sendo aplicada no desenvolvimento de projetos reais. Resultados dessa aplicação pode ser encontrado em [11].

6 Conclusão:

O esforço estimado é um insumo de fundamental importância para a elaboração dos cronogramas de projetos de desenvolvimento de software. Neste sentido, buscar melhorar continuamente a precisão dos resultados das estimativas conduz a maiores capacidades competitivas da organização e com maior segurança quanto ao cumprimento dos seus compromissos na entrega de seus produtos de software.

Este trabalho apresentou um conjunto de características que causam impactos nas produtividades de desenvolvimento de software e cujas análises podem levar a melhores analogias entre projetos já realizados e novos projetos a serem estimados. Além disso, as métricas estabelecidas mostram como a precisão pode ser avaliada conforme forem sendo executados os projetos dentro de uma abordagem de melhoria contínua do processo de estimativa de esforço. A descrição completa deste trabalho está em [28].

7 Referências Bibliográficas:

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

BASILI,V.,Rombach, H.Goal Question Metric Paradigm; Encyclopedia of Software Engineering – 2, 1994

BOEHM, Barry et al,Software Cost Estimation With COCOMO II, Prentice Hall PTR, 2000
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.
FUREY, Sean, Why we should use Function Points, IEEE, 1997
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
KOSLOSKI, R., OLIVEIRA, K.; An Experience Factory to Improve Software Development Effort Estimates; International Conference on Product Focused Sotfware Process Improvement – Profes 2005.
ISBSG. International Software Benchmarking Standards Group. The Benchmarking, Release 8, ISBSG; 2003.
JOHNSON, Corine N., The Benefits of PDCA, Quality Progress, pp.120, 2002
JONES, C. Software Challenges: Function point: a new way of looking at tools, Computer, August, 1994. p. 66 – 67.
LIM, Wayne C., Effects of Resue on quality, Productivity, and Economics, IEEE, 1994
LOKAN, Chris; TERRY, Wright; HILL, Peter R.; STRINGER, Michael; "Organizacional Benchmarking Using the ISBSG Data Repository", IEEE, 2001
LOW,Graham C., ROSS, D.Jeferry, "Function Points in the Estimation and Evaluation of the Software Process, IEEE, 1990
Martin, Arnold; PEDROSS, Peter; Software Size Measurement and Productivity Rating in Large Scale Softwre Development Department, IEEEE, 1998
MAXWELL, Katrina, FORSELIUS, Pekka, Benchmarking Software Development Productivity, IEEE, 2000
MCFEELEY, Bob, IDEAL: A User´s Guide for Software Process Improvement, SEI,1996
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
PSM, Pratical Software Measurement, Adison Wesley, 2002
RUBIN, Howard A ., Software Process Maturity: Measuring its Impact on Productivity and Quality, IEEE, 1993
SHEPPERD, Martin; CARTWRIGHT, Michele; Predicting with Sparse Data; IEEE, 2000
SPR, "The Programming Language Table", Software Productivity Research, 2001.

Kosloski; R. Melhoria Contínua de Estimativa de Esforço para o Desenvolvimento de Software: Uma abordagem sobre produtividade, Tese de Mestrado, Universidade Católica de Brasília, (a ser apresentado em abril de 2005)