PSPi – Melhorando Estimativas no Ambiente Corporativo

Sheila Reinehr[1, 2], Robert Burnett[1], Marcelo Pessôa[2]

[1]Pontifícia Universidade Católica do Paraná (PUCPR)
Rua Imaculada Conceição 1155 – Curitiba – PR – Brasil

[2]Escola Politécnica – Universidade de São Paulo (USP)
Av. Professor Almeida Prado, travessa 2, no. 128 – São Paulo - SP – Brasil

{sheila.reinehr,robert.burnett}@pucpr.br; mpessoa@usp.br

Abstract. *Although the information technology area has experienced several transformations along the last decades, supporting mainframe legacy systems is the reality of many organizations. At the same time, these companies wish to benefit from the new software engineering trends that confer more visibility and predictability to the software development. This paper presents an instance of the PSP (Personal Software Process) estimation method to be applied to the mainframe environment, using COBOL programming language.*

Resumo. *Apesar da área de tecnologia da informação ter passado por diversas transformações nas últimas décadas, manter sistemas legados em mainframe é a realidade de muitas organizações. Ao mesmo tempo, estas empresas desejam usufruir dos benefícios proporcionados pelas novas tendências da engenharia de software que conferem mais visibilidade e previsibilidade ao desenvolvimento de software. Este artigo apresenta uma instância do método de estimativas do PSP (Personal Software Process) para ser aplicada ao ambiente corporativo mainframe e aos projetos que utilizam a linguagem COBOL.*

1. Introdução

Nos últimos anos tem crescido a busca por uma maior formalização na área de engenharia de software, no sentido de torná-la efetivamente uma disciplina de engenharia. Em busca desta maior organização da área de tecnologia da informação, especialmente no campo de software, proliferaram modelos, normas, técnicas e novas ferramentas que são absorvidos e utilizados nas empresas.

Partindo do princípio que a qualidade do produto de software é fortemente influenciada pelo processo utilizado para produzi-lo [PAULK, 1994], diversos modelos que avaliam e auxiliam na melhoria dos processos de software começaram a focar a sua atenção sobre este aspecto. Alguns destes modelos são aplicáveis em nível organizacional, como os processos descritos na norma NBR ISO/IEC 12207, como o modelo CMMI – *Capability Maturity Model Integration* [SEI, 2006] ou o MPS.BR [SOFTEX, 2006]. Outros, são aplicáveis em nível pessoal ou de pequenos times, como o PSP – *Personal Software Process* [HUMPHREY, 1995], o TSP – *Team Software Process* [HUMPHREY, 1999], o RUP – *Rational Unified Process* [KRUCHTEN, 2000] e os métodos ágeis, como XP – *Extreme Programming*, EVO, SCRUM etc.

Em [GIBSON, 1997], o autor reconhece que à medida que as organizações amadurecem, ou seja, se aproximam do nível 3 do modelo SW-CMM (versão inicial do CMMI), descobrem que níveis superiores de maturidade são dependentes da melhoria do desempenho individual. Isso significa que não é suficiente que os processos organizacionais estejam claramente definidos e descritos, pois, em última instância, são pessoas que dão forma ao produto final: "Independentemente de quão bem uma organização é gerenciada, a qualidade do trabalho de software é governada pelas práticas dos engenheiros de software."

Levando-se em consideração: (i) o cenário atual que confirma as palavras de Grady Booch em [KRUCHTEN, 2000]: "... à medida que as empresas crescem e a tecnologia se torna disponível, cresce também a necessidade e a demanda por novos produtos e serviços"; (ii) o fato de que não é possível simplesmente desprezar todos os sistemas legados corporativos [SEACORD et al., 2001]; (iii) a importância das práticas individuais na melhoria organizacional [GIBSON, 1997], e; (iv) a dificuldade em se estimar tamanho e esforço para projetos que envolvem software; percebeu-se a necessidade de desenvolver uma abordagem para iniciar a melhoria de processos em equipes do ambiente *mainframe*, partindo-se da ótica individual.

Optou-se pela adaptação do modelo PSP (*Personal Software Process*) para o ambiente corporativo *mainframe*, como forma de oferecer uma primeira abordagem para a melhoria de processos. O PSPi, instância do PSP proposta por este trabalho, visa auxiliar as organizações que possuem ambiente de sistemas legados de grande porte, na tarefa de endereçar as questões relacionadas à estimativa de tamanho. Um bom processo de estimativas é a base para o início do amadurecimento do processo de planejamento, primeiro passo de qualquer modelo de melhoria organizacional, como o modelo CMMI [SEI, 2006] ou, no caso do Brasil, o MPS.BR [SOFTEX, 2006].

A seção 2 deste artigo descreve brevemente o modelo PSP; a seção 3 descreve o modelo proposto; a seção 4, apresenta os resultados da validação e a seção 5, estabelece as conclusões e os trabalhos futuros.

2. O PSP – *Personal Software Process*

O PSP é uma abordagem sistemática para o desenvolvimento de software que foi desenvolvida por Watts Humphrey e apresentada detalhadamente em [HUMPHREY, 1995], posteriormente simplificada em [HUMPHREY, 1997] e novamente refinada em [HUMPHREY, 2005]. Em 2005, foi lançado um corpo de conhecimento em PSP [POMMEROY-HUFF et al., 2005] que visa constituir a base da certificação profissional PSP, também lançada pelo SEI em 2005.

O PSP está estruturado em sete níveis de maturidade de processo através dos quais determinadas práticas de engenharia de software são introduzidas. O principal objetivo do PSP é mostrar aos engenheiros de software que um processo definido e mensurável pode auxiliá-los a melhorar o seu desempenho pessoal. À medida que o desempenho individual melhora, o desempenho de suas equipes e projetos também fica mais propício a melhorar [HUMPHREY, 1995].

Existem diversos relatos sobre a utilização do PSP acadêmica e industrialmente. Em [BOEHM & TURNER, 2003], os autores fazem uma boa análise do PSP em relação

a outras abordagens menos prescritivas, como os métodos ágeis, realçando quando é benéfico o emprego de métodos mais estruturados e formais como o PSP. Em [FERGUSON et al., 1998] os autores relatam o caso de três experiências industriais: *Advanced Information Services, Motorola Paging Products Group* e *Union Switch & Signal*. Em todas elas foram encontradas melhorias através do uso do PSP, quer na precisão das estimativas, quer na redução da densidade de defeitos.

Em [DELLIEN, 1997], Olof Dellien relata a introdução do PSP na *TXM - Empresa Tecnologica Ericsson*, em Saltillo-Mexico. Embora o trabalho não apresente resultados numéricos, diversas considerações são feitas no que diz respeito às adaptações necessárias ao contexto industrial e suas melhorias. Em [ESCALA & MORISIO, 1998] os autores discutem a introdução do PSP em uma empresa fabricante de controles numéricos italiana, a FIDIA, concluindo que o PSP é uma ferramenta útil para a melhoria individual, podendo ser implantado sem adaptações em alguns contextos, mas necessitando de customizações em outros. O *Centre for Software Engineering*, da Universidade de Dublin, apresenta em [CSE, 1998], dados referentes ao período de treinamento, e em seguida de utilização profissional, do PSP na *Advent Software*. Os números analisados apontam para uma redução efetiva no esforço despendido na fase de testes (alguns de 40% para 15% do total do desenvolvimento).

Disney e Johnson apresentam em [DISNEY, 1998] várias críticas em relação à qualidade dos dados coletados e analisados pelo PSP. O estudo foi baseado em 89 programas feitos por estudantes da Universidade do Hawaii. Apesar das críticas, os autores concluem que o PSP é uma ferramenta útil, desde que tomados os devidos cuidados com a coleta e análise dos dados e provida uma ferramenta automatizada. Em [PRECHELT & UNGER, 2001] os autores relatam os resultados da experiência da Universidade de Karlshure para validar os efeitos do treinamento em PSP, considerando 24 estudantes treinados em PSP e 16 não treinados. Entre outras conclusões, os autores constatam que o PSP é um método eficiente, porém, apenas o curso no formato padrão em que é oferecido não garante que seus participantes utilizarão as técnicas posteriormente. Este fato foi também constatado em experiências anteriores e, em certo grau, levou à criação do TSP [HUMPHREY, 1999], [HUMPHREY, 2006].

Em [BÖRSTLER et al., 2002] os autores apresentam as lições aprendidas com o ensino de PSP ao longo dos últimos anos e comparam as informações obtidas de cinco universidades americanas: Umea, Utah, Purdue, Montana Tech e Drexel.

No Brasil, o trabalho de Jones Albuquerque e Silvio Meira, descrito em [ALBUQUERQUE, 1997], propõe adaptações para a utilização do PSP com a linguagem JAVA, tecendo comparações com Smaltalk. O trabalho de Djalma Silva e Paulo Masiero, descrito em [SILVA, 1998], tem o seu foco nas adaptações efetuadas no modelo PSP para ser utilizado com o ambiente de desenvolvimento Powerbuilder, propondo modificações nos formulários e tabelas que são dependentes de linguagem. Em [SILVA, 1999], Marcos Silva e Mauro Spínola apresentam dados da utilização do PSP academicamente no Brasil, através de um modelo para introdução do PSP. Os resultados apontaram para uma redução média entre 25% e 36% na densidade total de defeitos.

3. PSPi – A instância proposta para o ambiente corporativo

3.1 Ambiente corporativo tradicional

Na maioria das organizações que possuem sistemas legados no ambiente *mainframe*, a linguagem padrão de terceira geração utilizada ainda é o COBOL. Evidências desse fato puderam ser observadas por ocasião da preparação para enfrentar o "*bug* do milênio", onde a demanda por desenvolvedores COBOL cresceu inesperadamente no mercado nacional e internacional. Isto nos remete a pensar na dicotomia levantada por Grady Booch [KRUCHTEN, 2000] e reforçada por Seacord et al. em [SEACORD et al., 2001], a respeito da necessidade de manter e evoluir os sistemas legados e adotar as novas tecnologias emergentes. Para essas organizações, continuar cumprindo seus objetivos de negócio, muitas vezes, significa continuar mantendo e evoluindo seus sistemas antigos em COBOL e outras linguagens de terceira geração, e passar a oferecer interfaces com o usuário desenvolvidas utilizando as novas tecnologias, linguagens e paradigmas. O sistema financeiro, por exemplo, em geral, possui os seus sistemas principais centralizados, processando de forma *batch*, em ambiente *mainframe*. Porém, seus produtos e serviços são oferecidos aos clientes através de interfaces mais amigáveis (Internet, celular, PDA etc.)

Em [SEACORD et al., 2001], ao discutir estratégias para a modernização de sistemas legados, os autores utilizam como estudo de caso um sistema de uma rede varejista composto de aproximadamente três milhões de linhas de código COBOL. Esforços como esse, além de onerosos e repletos de riscos, chegam a levar até seis anos para serem concluídos. Durante esse período, as atividades de manutenção continuam em andamento e continuarão, até que toda a modernização esteja concluída.

Como a instância que está sendo proposta, PSPi, se insere nesse contexto, o dos sistemas legados corporativos, todas as alterações que dependem de linguagem levaram em consideração a linguagem COBOL. Com essa abordagem, o trabalho aqui desenvolvido poderá ser utilizado de forma abrangente nessas empresas, cobrindo a maior parte do seu ambiente *batch* legado. A parte on-line do ambiente legado corporativo não será tratada por este trabalho devido à sua diversidade, em geral, composta por linguagens de quarta geração heterogêneas. Isso não significa que essa proposta não possa ser por eles utilizada, mas, sim, que adaptações terão que ser promovidas.

3.2 Visão geral do modelo proposto

A Figura 1 oferece uma visão geral das adaptações que se fizeram necessárias ao PSP para compor a instância proposta PSPi. O retângulo mais interno da figura (caixas brancas) representa o PSP original e o mais externo (caixas cinza), as adaptações desenvolvidas por este trabalho e denominadas, em seu conjunto, PSPi. É importante observar que as adaptações que estão sendo propostas por este trabalho diferem substancialmente, em abrangência e profundidade, das demais abordagens de outros trabalhos no Brasil, uma vez que afetam inclusive, e principalmente, o elemento central do PSP, o método de estimativas PROBE.

Conforme se pode observar pela Figura 1, os elementos que foram inseridos e/ou alterados para compor a instância PSPi são:

- Dois novos padrões: um para a codificação usando a linguagem COBOL, *C29i – COBOL Coding Standard* e outro para a contagem das LOC (Lines of Code), COBOL LOC *Counting Standard*;
- Um novo *checklist*: para apoiar a revisão de código de programas COBOL, *C58i - COBOL Code Review Checklist;*
- Um novo processo: para auxiliar a montagem da base de categorias de tamanho relativo, *PSPi0.2 – Size Categories Base*;
- Uma extensão ao método PROBE: um guia para auxiliar a estimativa de programas COBOL, C*36i - PROBEi Size Estimating Guidance*; e
- Uma alteração de formulário: gerando o formulário alterado *C39i – Size Estimating Template*, de modo a torná-lo aderente ao guia de estimativas proposto.

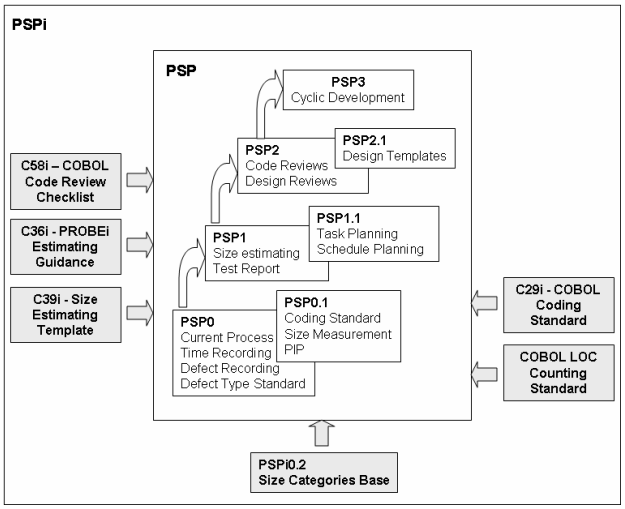

Figura 1. PSPi - Visão Geral do Modelo Proposto.

Com o objetivo de manter a compatibilidade com o PSP original, os elementos propostos por este trabalho, quando possível, seguem a nomenclatura e a numeração originais, seguidas da letra "i", representando a instância que está sendo proposta. Optou-se, ainda, por efetuar as amplificações em inglês, mais uma vez, para manutenção da compatibilidade com o original. Este artigo, apesar de mencionar todas as alterações realizadas no modelo, tem o seu foco no método de estimativas PROBE. Demais adaptações citadas podem ser encontradas na íntegra em [REINEHR, 2001].

3.3 PSPi novo elemento: C29i – COBOL Coding Standard

Foi proposto um padrão a ser utilizado para a codificação em linguagem COBOL, de modo a possibilitar maior consistência em relação às estimativas a serem realizadas. Nesse padrão buscou-se enfatizar dois aspectos que conferem facilidade de manutenção ao programa: documentação e clareza [REINEHR, 2001].

3.4 PSPi novo elemento: COBOL LOC Counting Standard

Estabelecer um padrão adequado para a contagem de linhas de código é muito importante para garantir que a medição possa ser reproduzida com os mesmos resultados [HUMPHREY, 1995]. O padrão proposto considera para a contagem apenas as linhas úteis da *Procedure Division.* As linhas das demais áreas do programa não devem ser contadas. Considerou-se como linha útil qualquer linha compreendida entre a sentença reservada *Procedure Division*, incluindo essa sentença, e o final do código. Foram desconsideradas, para efeito de linhas úteis, as linhas em branco, as linhas de comentários e as linhas que contém apenas o ponto de finalização (requerido pela linguagem).

3.5 PSPi novo elemento: PSPi0.2 - Size Categories Base

O elemento central do PSP é o método de estimativas PROBE – *Proxy Based Estimation*. Ele se baseia no conceito de *proxy*, uma abstração que permite ao engenheiro de software obter o tamanho em LOC do programa que ele está planejando desenvolver. Assim como o PROBE é o elemento central do PSP, as tabelas de categorias de tamanho relativo são os elementos centrais do PROBE. Através delas é que se consegue obter o tamanho relativo em LOC/método de um objeto, conforme ilustram as tabelas propostas por Humphrey para C++ e Object Pascal [HUMPHREY, 1995] ou a proposta por [SCHOEDEL, 2006] para a linguagem SQL. No entanto, quando o desenvolvedor inicia a utilização do PSP, além de ele normalmente não dispor de sua própria base de categorias, ele também não sabe como montá-la.

Optou-se, neste trabalho, por criar um novo nível intermediário de processo, o *PSPi0.2*. Esse nível utiliza alguns elementos propostos no Anexo A de [HUMPHREY, 1995] e incorpora novos elementos, auxiliando o engenheiro de software, e, em última instância a própria empresa, a montar sua base de categorias inicial. A Figura 2 ilustra o processo *PSPi0.2* proposto. Como o PROBE passa a ser utilizado a partir do nível PSP1 e a sua utilização pressupõe a existência das categorias de tamanho, optou-se por situar o nível intermediário *PSPi0.2* antes do nível PSP1. A descrição detalhada do processo proposto PSPi0.2 pode ser encontrada em [REINEHR, 2001].

Conforme se pode observar na Figura 2, o primeiro passo proposto é identificar, a partir do domínio da aplicação, os tipos de programas que farão parte da base de categorias. No âmbito das aplicações comerciais que processam no ambiente *mainframe* pode-se considerar três tipos básicos de programa:

- Receptor: programa que recebe dados diretamente do usuário final, com a finalidade ou não de armazenamento. No ambiente corporativo de grande porte, em geral, esses programas são desenvolvidos usando linguagens de quarta geração ou linguagens de terceira geração associadas a serviços de um monitor transacional. Representam as entradas on-line do sistema.
- Atualizador: programa *batch* que, em geral, faz o processamento e atualização de grandes volumes de informação, envolvendo o negócio da aplicação propriamente dito, atualizando as principais bases da aplicação.

- Formatador: programa *batch* que faz a extração de informações das bases da aplicação e as disponibiliza para entidades externas, quer na forma de consulta on-line, relatório impresso, ou outro meio magnético qualquer.

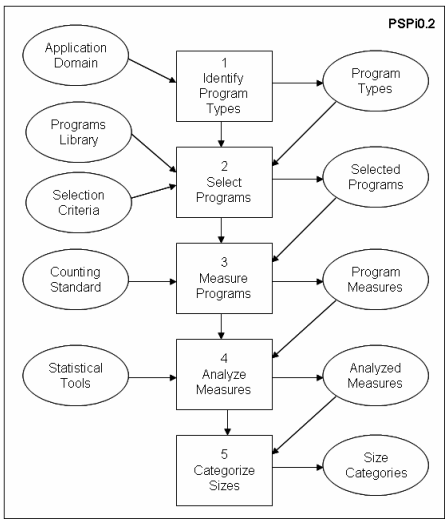

Figura 2. *PSPi0.2 - Size Categories Base.*

Conforme citado anteriormente, não é foco deste trabalho tratar a porção on-line dos ambientes corporativos de grande porte devido à sua variedade. Portanto, os programas do tipo Receptor (em geral telas de entrada de dados do ambiente on-line) não serão considerados, sendo tratados apenas os programas dos tipos: Formatador e Atualizador.

O resultado final que se obtém do processo *PSPi0.2* é a tabela de categorias de tamanho relativo a ser utilizada nas estimativas de tamanho do método PROBE. Essas tabelas fornecerão o tamanho médio, em LOC/*section*, de cada uma das categorias de tamanho de *section*: muito pequena (VS), pequena (S), média (M), grande (L) e muito grande (VL). Juntamente com o guia para as estimativas descrito na próxima seção, esta tabela de categorias constitui uma das principais contribuições deste trabalho.

3.6 PSPi novo elemento: C36i - PROBEi Size Estimating Guidance

Ao iniciar a utilização do método PROBE do PSP original, a primeira dificuldade que o engenheiro de software enfrenta é a elaboração conceitual do programa e, em seguida, a seleção da categoria de tamanho relativo mais adequada para cada *proxy*. É possível que, mesmo tendo experiência, ele não saiba como estimar tamanho de programa ou até mesmo não saiba como produzir um projeto conceitual. Como o método PSP original não provê qualquer tipo de ajuda nesse sentido, optou-se, na instância PSPi, por desenvolver um conjunto de orientações, agrupadas na forma de um guia (*C36i – PROBEi Size Estimating Guidance*).

A Figura 3 apresenta, de forma resumida, as orientações que foram elaboradas para auxiliar o desenvolvedor na tarefa de planejar o *design* e a estimativa para o programa.É importante ressaltar que este guia não tem o objetivo de substituir a

experiência do profissional, mas de complementá-la e de servir como ponto de partida para o seu aprendizado. Para a versão completa deste guia, consultar [REINEHR, 2001].

SECTION	DESCRIPTION	REL SIZE
Main section	Consider one main section for each program.	S
Starting section	Consider one starting section for each program. Its size may vary according to the amount of parameter initializations needed.	M
Normal ending section	Consider one normal ending section for each program.	S
Abnormal ending section	Consider one abnormal ending section for each program.	S
Open section (Any kind of file)	Consider one open section for each input file, input database table, output file and output database table.	S
Close section (Any kind of file)	Consider one close section for each input file, input database table, output file and output database table.	S
Read section (sequential and VSAM)	Consider at least one read section for each input file.	S
Read section (database tables)	Consider at least one read section for each read only database table.	M
Write section (sequential files, VSAM files and report files)	Consider at least one write section for each output file and for each input-output file. Consider at least one write section for each type of record.	S
Write section (database tables)	Consider at least one write section for each updateable database table.	M
External call section (subroutines)	Consider one external call section for each different subroutine. The size may vary according to the amount of entry parameters to be set.	S to M
Report control section	Consider one report control section.	S
Selection section	One S section usually holds 3 selection conditions, considering one IF command for each selection criteria.	3 / S
Calculation section	One S section usually holds 5 calculation commands, considering one COMPUTE or equivalent command for each calculation need.	5 / S
Update Logic section	One S section usually holds 15 MOVE or equivalent commands.	15 / S
Classification section (using internal SORT)	Either the input or the output sort section may vary from VS to VL. Generally, the complexity is located in one of them.	VS to VL

Figura 3. C36i - PROBEi Size Estimating Guidance.

3.8 PSPi novo elemento: C58i - COBOL Code Review Checklist

Um *checklist* para a revisão de programas pode ser considerado adequado quando auxilia o desenvolvedor a encontrar, não apenas o maior número de erros presentes no código, como também o maior número de potenciais problemas de processamento, no menor espaço possível de tempo. Embora alguns elementos do *checklist* de revisão de código sejam genéricos, ou seja, independam da linguagem que se está revisando, a maioria é bastante aderente ao ambiente em questão, conforme demonstrado pelos trabalhos de [ALBUQUERQUE, 1997], [SILVA, 1999] e [SCHOEDEL, 2006], que alteram as propostas de *chekclist* do PSP original para instanciação para os seus ambientes Java, Powerbuilder e SQL. O formulário proposto no PSPi para ser utilizado como guia para revisão de código COBOL é o *C58i – COBOL Code Review Checklist* que pode ser visto em detalhes em [REINEHR, 2001].

3.7 PSPi novo elemento: C39i – Size Estimating Template

O formulário *C39 - Size Estimating Template* do PSP original foi alterado, gerando uma nova versão (*C39i – Size Estimating Template*) que substitui as seções *Base Additions* e *New Objects*. A Figura 4 mostra apenas os trechos substituídos. O formulário completo pode ser consultado em [REINEHR, 2001].

BASE ADDITIONS	VS	S	M	L	VL
MAIN SECTION (S)					
STARTING SECTION (M)					
NORMAL ENDING SECTION (S)					
ABNORMAL ENDING SECTION (S)					
OPEN SECTION (S)					
CLOSE SECTION (S)					
READ SECTION - VSAM/SEQ (S)					
READ SECTION – DB (M)					
WRITE SECTION – VSAM/SEQ (S)					
WRITE SECTION – DB (M)					
EXTERNAL CALL SECTION (S to M)					
REPORT CONTROL SECTION (S)					
SELECTION SECTION (3/S)					
CALCULATION SECTION (5/S)					
UPDATE SECTION (15/S)					
CLASSIFICATION SECTION (VS to VL)					
OTHER SECTIONS (VS to VL)					
NUMBER OF SECTIONS					
LOC/SECTION					
TOTAL LOC/SECTION					
TOTAL BASE ADDITIONS (BA)	=>	=>	=>	=>	

NEW SECTIONS	VS	S	M	L	VL
MAIN SECTION (S)					
STARTING SECTION (M)					
NORMAL ENDING SECTION (S)					
ABNORMAL ENDING SECTION (S)					
OPEN SECTION (S)					
CLOSE SECTION (S)					
READ SECTION - VSAM/SEQ (S)					
READ SECTION – DB (M)					
WRITE SECTION – VSAM/SEQ (S)					
WRITE SECTION – DB (M)					
EXTERNAL CALL SECTION (S to M)					
REPORT CONTROL SECTION (S)					
SELECTION SECTION (3/S)					
CALCULATION SECTION (5/S)					
UPDATE SECTION (15/S)					
CLASSIFICATION SECTION (VS to VL)					
OTHER SECTIONS (VS to VL)					
NUMBER OF SECTIONS					
LOC/SECTION					
TOTAL LOC/SECTION					
TOTAL NEW SECTIONS (NO)	=>	=>	=>	=>	

Figura 4. C39i - Size Estimating Template – substituições.

4. Validação dos Resultados

Como forma de validar as propostas efetuadas por este trabalho, uma aplicação prática do método foi conduzida, tendo sido realizada em duas etapas. A primeira, consistiu na execução do novo processo proposto pelo PSPi, o processo *PSPi0.2 Size Categories Base*, para a montagem da base de categorias de tamanho. A segunda, consistiu na aplicação da base de categorias de tamanho desenvolvida, em conjunto com as orientações para a concepção e estimativa de programas proposta através do *C36i - PROBEi Size Estimating Guidance.*

4.1. Ambiente de inserção

A aplicação prática desta proposta foi realizada em um ambiente de grande porte de uma instituição financeira cuja porção batch dos sistemas de informação legados é, na sua maioria, codificada em linguagem COBOL, com ou sem acesso direto a informações em banco de dados utilizando SQL. Os programas utilizados na primeira etapa da validação (para a montagem da base de categorias de tamanho) são provenientes de um Sistema de

Controle de Financiamentos. Ao todo foram mensurados e catalogados 130 programas batch COBOL, sendo 100, da categoria Formatador e 30, da categoria Validador.

Os programas utilizados para a segunda etapa da validação (validação da base de categorias de tamanho e das orientações) são provenientes de um Sistema de Pagamentos do sistema financeiro. Ao todo foram estimados, mensurados e catalogados 10 programas da categoria Formatador e 15 programas da categoria Atualizador, representando a totalidade dos programas batch COBOL da aplicação.

Como forma de reforçar a independência dos resultados, a segunda etapa foi conduzida por um profissional de mercado, experiente em sistemas de grande porte, sem vínculo com as etapas anteriores de concepção do método.

4.2. Primeira etapa da validação: montagem da base de categorias de tamanho

A base de categorias de tamanho foi montada pela autora desse trabalho, seguindo o processo PSPi0.2 (Figura 2), que foi proposto pela instância PSPi, através das seguintes atividades:

- Identificação dos tipos de programas: Os programas do tipo on-line que utilizam linguagem de quarta geração proprietária do banco de dados e/ou linguagem COBOL associada ao monitor transacional CICS, no caso, COBOL CICS, não serão tratados por este trabalho. Serão considerados os programas *batch* escritos em COBOL, das categorias Formatador e Atualizador.
- Seleção do conjunto de programas: Para garantir a representatividade dos programas selecionados em relação ao universo que se desejava atender, foram escolhidos programas de uma aplicação que pertence ao segmento da linha de frente de negócios da empresa (sistema de financiamentos de máquinas e equipamentos agro-industriais), cuja parte batch é composta por 95% de programas COBOL, garantindo que as diversas categorias de programa são representadas na amostra. Ao todo foram selecionados 100 (cem) programas da categoria Formatador e 30 (trinta), da categoria Atualizador. Pode-se portanto, para fins estatísticos, considerar que se está trabalhando com toda a população de cada uma das categorias.
- Mensuração dos programas selecionados: Os programas selecionados foram mensurados utilizando as regras de contagem estabelecidas.
- Análise da base de programas: As categorias de programas foram testadas em relação à normalidade da distribuição pelo teste gráfico e do $\chi 2$. O valor obtido no teste de $\chi 2$ foi de 0,10879 para a categoria Formatador e 0,44077 para a categoria Atualizador; o que reflete uma aproximação da distribuição normal.
- Montagem das categorias de tamanho do proxies: Como os testes realizados acima apontaram para distribuições normais, definiu-se como categorias de tamanho listadas na Figura 5.

	MUITO PEQUENO (VS) $\mu - 2 * \sigma$	PEQUENO (S) $\mu - \sigma$	MÉDIO (M) μ	GRANDE (L) $\mu + \sigma$	MUITO GRANDE (VL) $\mu + 2 * \sigma$
Formatador	9,242	19,842	30,441	41,041	51,640
Atualizador	10,479	17,062	27,782	45,238	73,661

Figura 5. Categorias para estimativas de programas COBOL.

4.3. Segunda etapa: validação da base de categorias de tamanho e orientações

Uma vez montadas as duas bases de categorias de tamanho relativo de programa COBOL, foi necessário validar se as orientações propostas por este trabalho, combinadas com essas bases, resultariam em um bom método de estimativas. Conforme citado anteriormente, de modo a reforçar a independência dos resultados, essa validação foi efetuada por uma analista de sistemas, com experiência em desenvolvimento e manutenção de sistemas no ambiente *mainframe.* Ainda para reforçar a independência, foi selecionado um novo conjunto de programas de outro sistema, para serem estimados (10 da categoria Formatador e 15 da categoria Atualizador).

Foi fornecido para a analista que efetuaria as estimativas: o guia de estimativas *C36i – PROBEi Size Estimating Guidance* desenvolvido por este trabalho; as bases de categorias de tamanho relativo dos *proxies* dos programas do tipo Formatador e Atualizador desenvolvidas na primeira etapa da validação, utilizando as orientações propostas por este trabalho através do processo PSPi02; o formulário de estimativas *C39i Size Estimating Template* cujas complementações foram propostas por este trabalho. Essa segunda etapa da validação consistiu de duas atividades:

- A analista de sistemas seguiu as orientações do guia *C36i – PROBEi Size Estimating Guidance,* utilizou as bases de categorias de tamanho relativo, e estimou o tamanho de cada um dos programas componentes do sistema de pagamentos;
- Os programas reais do sistema de pagamentos foram mensurados, para que fosse determinado o número real de LOC, e os resultados da estimativa foram comparados com esses dados reais.

4.4 Estimativa de tamanho utilizando o método proposto

Para iniciar as estimativas de tamanho a analista de sistemas dispunha, além de sua experiência em análise, das informações básicas sobre os requisitos do programas. Ela não tinha experiência anterior em estimar LOC. Primeiramente, a analista de sistemas identificou a quantidade de *sections* necessárias para cada programa e qual o tamanho relativo mais adequado de cada *section* (VS, S, M, L ou VL), registrando essas informações no formulário *C39i Size Estimating Template.* Em seguida, utilizando a base de categorias de tamanho relativo da categoria Formatador e da categoria Atualizador, calculou o tamanho total de cada programa.

4.5 Análise das estimativas

Uma vez concluída a etapa de estimativas, os programas foram submetidos à mensuração utilizando uma rotina de contagem de LOC seguindo as regras estabelecidas no padrão de contagem de LOC do método.

Os resultados obtidos com a mensuração dos programas do tipo Formatador são apresentados no gráfico ao lado esquerdo da Figura 6. Conforme se pode observar, o método de estimativas para essa categoria tende a produzir estimativas a maior. Apenas no programa número 1, que possuía características um pouco diferentes dos demais, a estimativa foi a menor. Sendo assim, considerando os demais programas, o método parece induzir o desenvolvedor a produzir estimativas de tamanho superiores aos tamanhos reais.

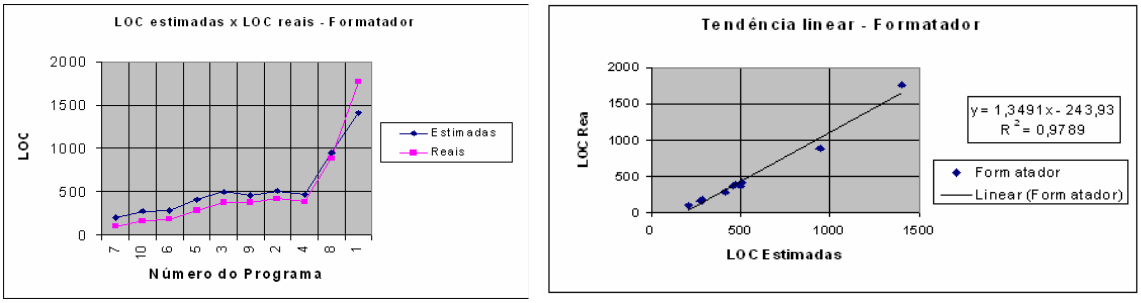

Figura 6. Resultados do experimento na Categoria Formatador.

O gráfico do lado direito da Figura 6 apresenta os resultados na forma de dispersão, incluindo a linha de tendência linear e a sua respectiva equação linear. Esta poderá ser utilizada posteriormente nas próximas estimativas, como forma de adequar os desvios naturais da estimativa. Como se pode observar, a correlação entre as LOC estimadas utilizando o método e as LOC reais para os programas do tipo Formatador é bastante alta, $r^2 = 0{,}9789$.

Analogamente, o gráfico da esquerda da Figura 7 apresenta as LOC estimadas e LOC reais para os 15 programas do tipo Atualizador. Conforme se pode observar, não há uma tendência otimista ou pessimista predominante, havendo estimativas a maior e a menor, em proporção similar. Esse pode ser um indicativo de que o método para o tipo Atualizador é mais equilibrado em relação à tendência de erro do que o método do tipo Formatador.

O lado direito da Figura 7 ilustra as informações em um gráfico de dispersão, também com a respectiva equação linear. O grau de correlação existente nessa categoria de programas é menor, no entanto é também bastante alto, sendo r^2=0,829.

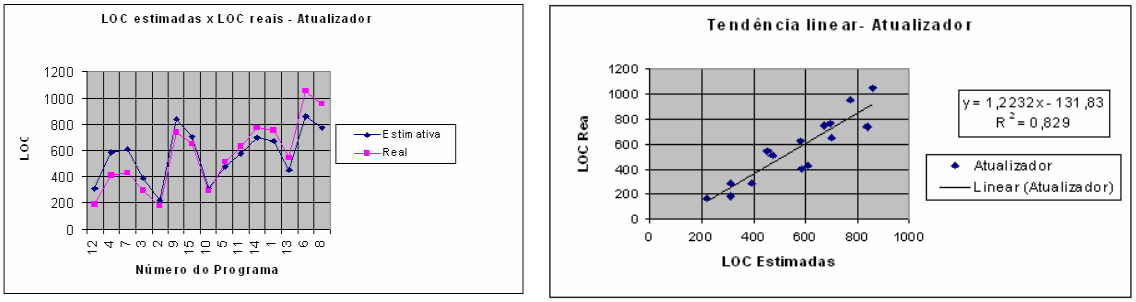

Figura 7. Resultados do experimento na Categoria Atualizador.

4.6 Análise dos resultados

Os resultados surpreenderam os autores que não esperavam obter uma estimativa tão próxima da realidade e com um grau de correlação tão elevado. Conforme demonstrado nas seções anteriores, a instância PROBEi, proposta dentro do modelo PSPi, mostrou-se adequada para a estimativa de programas do ambiente corporativo, utilizando a linguagem COBOL.

Os programas utilizados para a montagem dessa base não foram desenvolvidos por uma única pessoa, podendo conter desvios naturais de tamanho devido ao uso de diferentes estilos de programação. Não se considerou, no entanto, para o contexto deste trabalho, que isto pudesse constituir uma diferença significativa. Primeiro, porque a linguagem utilizada, o COBOL, não permite as inúmeras variações que outros tipos de linguagem, cuja estrutura é menos rigorosa, permitiriam (Ex.: C, C++, JAVA.). Segundo, porque as orientações propostas no guia *C36i – PROBEi Size Estimating Guidance* ajudam a reduzir possíveis variações, uma vez que levam à estimativa de cada necessidade do programa, independentemente do estilo de programação adotado.

As bases de categorias que foram desenvolvidas poderão, e deverão, ser melhoradas pelo próprio desenvolvedor, à medida que ele for incorporando a elas os seus próprios programas, auxiliando a calibrar a tendência linear. Além disso, o próprio guia com as orientações para a estimativa poderá ser melhorado à medida que o desenvolvedor analise o motivo dos desvios nas estimativas e incorpore novas regras.

A segunda etapa da aplicação prática foi conduzida com o objetivo de validar as bases de categorias montadas e as alterações propostas na instância PSPi. Nessa etapa, buscou-se o maior grau de independência possível para a realização das estimativas. Primeiramente, selecionando para executá-las uma profissional técnica que não participou de qualquer uma das etapas anteriores deste trabalho. Segundo, selecionando para a validação uma aplicação completa, sem a exclusão de qualquer programa *batch* COBOL da amostra.

5. Conclusões e Trabalhos Futuros

Através de suas extensões ao modelo PSP, a instância proposta PSPi, oferece uma importante contribuição para as empresas que estão implantando melhoria de processos. Ela apresenta uma forma de auxiliar o desenvolvedor na tarefa de estimar o tamanho do programa que será desenvolvido, uma das etapas críticas do processo de planejamento. As bases de categorias de tamanho relativo de *proxies*, apresentadas separadamente para os programas do tipo Formatador e Atualizador, comprovaram ser úteis no planejamento do tamanho quando combinadas com as orientações para a concepção do programa, descritas na extensão *C36i – PROBEi Size Estimating Guidance.*

Este trabalho apresenta contribuições relevantes ao modelo individual de melhoria de processos PSP, no entanto, ainda há o que evoluir e contribuir para o amadurecimento desta e de outras práticas da engenharia de software, como: a incorporação da instância PSPi ao modelo TSP [HUMPHREY, 2006]; a extensão da instância PSPi para o tratamento dos programas do tipo Receptor, utilizando diferentes linguagens de quarta geração de mercado; a incorporação no PSPi de alterações que tratem as fases iniciais do ciclo de vida, especificamente a eliciação de requisitos, não tratada extensivamente nem pelo PSP, nem pelo PSPi.

Conforme citado pelos autores em [REINEHR, 2000]: “O conhecimento é a matéria prima para o desenvolvimento de software, e é o engenheiro de software quem transforma conhecimento em produtos de software. Portanto, não se pode negligenciar as práticas individuais quando se deseja implementar um modelo de melhoria de processos nas organizações, seja qual for o seu porte ou nacionalidade. O PSP, se não

perfeito, é um bom caminho para a indústria brasileira iniciar sua jornada rumo aos níveis internacionais de qualidade."

Referências

[ALBUQUERQUE, 1997] ALBUQUERQUE, J.; "PSP-JOA Processo de Software Pessoal – Uma abordagem Orientada a JAVA". Dissertação de Mestrado. Recife: UFPE, 1997, 141p.

[BOEHM & TURNER, 2003] BOEHM, B.; TURNER, R. "Balancing Agility and Iterative Development: A Guide for the Perplexed". Reading, MA: Addison-Wesley, 2003, 266p.

[BÖRSTLER et al., 2002] BÖRSTLER, J.; CARRINGTON, D.; HISLOP, G.; LISACK, S.; OLSON, K.; WILLIAMS, L.. "Teaching PSP: Challenges and Lessons Learned". IEEE Software, 19(5), pp. 42-48, 2002.

[CSE, 1998] CSE – Centre for Software Engineering. "Software Process Improvement – Case Study". Dublin: Spire Book, Número 5, November 1998.

[DELLIEN, 1997] DELLIEN, O. "The PSP in Industry." Tese de Mestrado. Lund: Lund Institute of Technology – Department of Communication Systems, 1997, 41p.

[DISNEY, 1998] DISNEY, A. "Data Quality Problems in the PSP". Tese de Mestrado. Honolulu: University of Hawaii, 1998, 95p.

[ESCALA & MORISIO, 1998] ESCALA, D.; MORISIO, M. "A metric suite for a team PSP". In proceedings of The Fifth International Software Metrics Symposium, Bethesda, Maryland, pp. 61-71, 1998.

[FERGUSON et al., 1998] FERGUSON, P.; HUMPHREY, W.; KHAJENOORI, SOHEIL; M., S.; MATVYA; A. "Introducing the Personal Software Process: Three industrial cases". IEEE Computer, Volume 30, Número 5, pp. 24-31, Maio 1998.

[GIBSON, 1997] GIBSON, R. "Applied Software Process Improvement". In proceedings of the Americas Conference on Information Systems, pp. 596-598, 1997.

[HUMPHREY, 1995] HUMPHREY, W. "A Discipline for Software Engineering". Reading, MA: Addison-Wesley, 1995, 789 p.

[HUMPHREY, 1997] HUMPHREY, W. "Introduction to the Personal Software Process". Reading,MA: Addison-Wesley, 1997, 278p.

[HUMPHREY, 1999] HUMPHREY, W. "Introduction to the Team Software Process". Reading, MA: Addison-Wesley, 1999, 463 p.

[HUMPHREY, 2005] HUMPHREY, W. "PSP – A Self Improvement Process for Software Engineers". Upper Saddle River, NJ: Addison-Wesley, 2005, 346p.

[HUMPHREY, 2006] HUMPHREY, W. "TSP – Leading a Development Team". Upper Saddle River, NJ: Addison-Wesley, 2006, 307p.

[JACOBSON, BOOCH & RUMBAUGH, 1999] JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. "The Unified Software Development Process". Reading, MA: Addison-Wesley, 1999, 463 p.

[KRUCHTEN, 2000] KRUCHTEN, P. "The Rational Unified Process – An Introduction". 2nd Ed. Reading, MA: Addison-Wesley, 2000, 298 p.

[PAULK, 1994] PAULK, M.; CURTIS, B.; CHRISSIS, M.; WEBER, C. "Capability Maturity Model for Software: Guidelines for Improving the Software Process." Reading,MA: Addison-Wesley, 1994, 441 p.

[POMMEROY-HUFF et al., 2005] POMMEROY-HUFF, M.; MULLANEY, J.; CANNON, R.; SEBERN, M. "The Personal Software Process (PSP) Body of Knowledge – Version 1.0 (CMU/SEI-2005-SR-003)". Pittsburgh: Software Engineering Institute, 2005, 74 p.

[PRECHELT & UNGER, 2001] PRECHELT, L.; UNGER, B. "An Experiment Measuring the Effects of Personal Software Process (PSP) Training". IEEE Transactions on Software Engineering, 27(5), pp. 465-472, 2001.

[REINEHR, 2000] REINEHR, S.; BURNETT, R. "PSP: uma boa opção para a indústria brasileira?" Anais da XI CITS Conferência Internacional de Tecnologia de Software: Qualidade de Software, Curitiba-PR, pp. 85-96, 2000.

[REINEHR, 2001] REINEHR, S. "PSPi – Uma instância do Personal Software Process para o ambiente corporativo". Dissertação de Mestrado. Curitiba: Pontifícia Universidade Católica do Paraná, 2001, 145p.

[SCHOEDEL, 2006] SCHOEDEL, R. PROxy Based Estimation (PROBE) for Structured Query Language (SQL) (CMU/SEI-2006-TN-017). Pittsburgh: Software Engineering Institute, 2006, 22 p.

[SEACORD et al., 2001] SEACORD, R.; DORDA-COMELLA, S.; LEWIS, G.; PLACE, P.; PLAKOSH, D. "Legacy System Modernization Strategies". (CMU/SEI-2001-TR-025). Pittsburgh, PA: SEI, Carnegie Mellon University, 2001.

[SEI, 2006] SOFTWARE ENGINEERING INSTITUTE. "CMMI for Development - V1.2 (CMU/SEI-06TR008)". Pittsburgh: Software Engineering Institute, 2006.

[SILVA, 1998] SILVA, D. "Uma Proposta de Versão e Ferramenta do PSP para o Domínio de Aplicações Comerciais". Anais da IX CITS Conferência Internacional de Tecnologia de Software: Qualidade de Software, Curitiba-PR, pp. 157-168, 1998.

[SILVA, 1999] SILVA, M. "Um método para a utilização do modelo PSP". Dissertação de Mestrado. São Paulo: UNIP, 1999.

[SOFTEX, 2006] SOCIEDADE PARA A EXCELÊNCIA DO SOFTWARE BRASILEIRO. "MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral (v1.1)". Campinas: SOFTEX, 2006, 56p.