Proposta e Avaliação de um Processo para Agrupamento de Solicitações de Manutenções

Gladston J. Aparecido[1], Marcelo Nassau Malta[2], Humberto Mossri de Almeida[2], Humberto T. Marques-Neto[1,2], Marco Túlio Valente[3]

[1]Instituto de Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
30.535-610 - Belo Horizonte - Brasil

[2]Divisão de Tecnologia (DATAPUC)
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
30.150-220 - Belo Horizonte - Brasil

[3]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
31.270-010 - Belo Horizonte - Brasil

gladston.aparecido@sga.pucminas.br, mtov@dcc.ufmg.br

{nassau,hmossri,humberto}@pucminas.br

Resumo. *O planejamento e gerenciamento da manutenção de software é um problema tanto para usuários quanto para engenheiros de software. O processo que conduz a execução das demandas registradas pelos usuários pode interferir diretamente no custo da manutenção. O presente estudo apresenta o Processo para Agrupamento de Solicitações de Manutenção (PASM), descrevendo suas diretrizes para guiar a priorização, agrupamento e execução de diversos tipos de manutenção de software. O PASM foi aplicado na Divisão de Tecnologia da PUC Minas e entre os benefícios identificados encontram-se o aumento de 18,76% das horas trabalhadas em projetos de manutenção e um declínio de 27,73% nas horas trabalhadas em solicitações urgentes.*

Abstract. *The planning and management of software maintenance is a problem for both users and software engineers. The process that guides the implementation of maintenance requests can have a direct impact in software costs. This paper presents a Process for Batching Maintenance Requests (PASM, in Portuguese), describing its guidelines to prioritization, batching, and implementation of various types of software maintenance requests. PASM has been applied in the Technology Division of PUC Minas. Among the main benefits derived from its implementation, we can mention an increase of 18.76% in the hours allocated to maintenance projects and a decline of 27.73% in the hours allocated to urgent maintenance requests.*

1. Introdução

Manutenção representa uma das fases mais importantes e onerosas do ciclo de vida de um sistema. Por exemplo, existem estudos que mostram que até 90% do total de recursos consumidos ao longo da vida útil de um sistema são destinados à fase de manuten-

ção [Erlikh 2000]. Outros estudos calculam que pelo menos 250 bilhões de linhas de código encontravam-se em manutenção no mundo no início da década [Sommerville 2004]. Por fim, em meados da década de 90, Sutherland estimou em US$ 70 bilhões os gastos anuais que empresas norte-americanas tinham com atividades de manutenção de software [Sutherland 2000]. Nos últimos anos, nada indica que estes custos tenham diminuído. Na verdade, novos sistemas são desenvolvidos a cada dia, para os mais diversos domínios de aplicação. Com isso, cria-se uma base crescente de código que tem que ser continuamente mantida e evoluída.

Apesar de sua inegável importância, atividades de manutenção normalmente têm uma imagem negativa entre os membros da equipe de manutenção, incluindo seus gestores [Tan and Mookerjee 2005]. Como resultado, existe uma tendência em não tratar manutenções como projetos de software, isto é, projetos que – tal como ocorre no caso de novos sistemas – possuem atividades de definição de requisitos, análise, implementação, testes, etc. Em vez disso, é comum considerar solicitações de manutenção como atividades de curta duração, destinadas a resolver demandas urgentes e que, portanto, devem ser implementadas rapidamente, de forma a não impactar as atividades dos usuários finais dos sistemas em questão. Particularmente, esse tratamento contínuo das solicitações de manutenção é mais comum no caso de sistemas desenvolvidos internamente pelos departamentos de Tecnologia da Informação das empresas, visto que nesse caso os usuários internos estão mais próximos dos desenvolvedores e gerentes de projeto (e, em alguns casos, possuem um nível hierárquico superior ao desses últimos).

Os problemas inerentes ao tratamento contínuo e individual de solicitações de manutenção já foram apontados em alguns estudos. Por exemplo, Banker e Slaughter investigaram os ganhos de escala que são aferidos quando as manutenções são tratadas em modo *batch*, isto é, quando solicitações individuais são empacotadas e implementadas em projetos de maior tamanho [Banker and Slaughter 1997]. Segundo eles, essa estratégia pode reduzir os custos de manutenção em 36%. Os benefícios de manutenções periódicas – combinadas com políticas para substituição de um sistema – foram também apontados por Tan e Mookerjee [Tan and Mookerjee 2005]. No entanto, esses trabalhos se valem de modelos analíticos para recomendar o tratamento de manutenções como projetos de software. Ou seja, os benefícios apontados não foram comprovados em ambientes reais de desenvolvimento de software. Além disso, as recomendações contidas nesses trabalhos são, bastante genéricas, visto que, eles não propõem um fluxo detalhado de atividades destinadas a transformar solicitações contínuas de manutenção em projetos de software.

Este artigo apresenta um Processo para Agrupamento de Solicitações de Manutenções de software, chamado PASM. Basicamente, esse processo propõe um fluxo detalhado de atividades destinadas a agrupar solicitações de manutenção individuais em projetos de software. Uma versão preliminar do PASM foi apresentada em um artigo resumido de *workshop* [Malta et al. 2009]. Nesse primeiro artigo, as ideias gerais do processo foram enunciadas e os primeiros resultados de sua implantação foram apresentados. O presente artigo complementa esse primeiro trabalho, especificando as etapas do PASM. Descreve-se também a implantação do processo na Divisão de Tecnologia da PUC Minas (DATAPUC). Particularmente, no DATAPUC, até o final de 2008, as manutenções eram implementadas à medida que eram solicitadas pelos usuários, com graus de priorização e empacotamento definidos de forma *ad hoc* para cada sistema. A partir de 2009, o DATA-

PUC passou a adotar o processo para agrupamento de solicitações proposto neste trabalho. Como resultado, observou-se um aumento de 18,76% das horas trabalhadas em projetos de manutenção e um declínio de 27,73% nas horas trabalhadas em solicitações urgentes, acompanhado de uma melhoria significativa da produtividade da equipe de manutenção.

O restante deste artigo está organizado do seguinte modo. Na Seção 2, descreve-se o PASM, incluindo uma apresentação de seus objetivos, dos princípios que guiaram a sua proposição e das principais etapas propostas por este processo. Na Seção 3, descreve-se a implantação do PASM na Divisão de Tecnologia da PUC Minas. Basicamente, para se avaliar os benefícios e eventuais limitações do processo, comparam-se diversos indicadores de qualidade do DATAPUC referentes ao período após a implantação do PASM com os seus valores antes da implantação do processo. A Seção 4 apresenta os principais trabalhos relacionados com esta pesquisa. Por fim, a Seção 5 conclui o artigo e apresenta linhas para trabalhos futuros.

2. Processo para Agrupamento de Solicitações de Manutenções de Software

O Processo para Agrupamento de Solicitações de Manutenções de Software (PASM) está fundamentado em duas premissas essenciais: (1) as demandas de manutenção devem estar organizadas em projetos para maximizar a eficiência das atividades de planejamento e controle de execução e (2) o registro de solicitações das áreas usuárias deve ser feito de forma disciplinada e organizada para viabilizar a elaboração de propostas de projetos e, consequentemente, diminuir a quantidade de manutenções classificadas como urgentes.

O PASM é executado em um ciclo mensal de elaboração, avaliação e aprovação de propostas de projetos de manutenção de software. Cada ciclo do processo é constituído por dez etapas realizadas ao longo de quatro semanas. Com o propósito de se estabelecer um relacionamento do período de registro de solicitações de manutenções com o começo de um determinado mês, recomenda-se que o ciclo mensal seja iniciado na última semana do mês anterior. A seguir, apresenta-se uma visão geral das etapas do PASM:

1. **Reuniões de planejamento:** Na última semana de cada mês, os analistas de sistemas e gerentes de projetos devem realizar reuniões com os usuários-chave para apresentar a situação de cada projeto de manutenção que se encontra em execução e discutir as demandas, novas ou antigas, que deverão ser avaliadas no ciclo de priorização de projetos de manutenção de software do mês subseqüente.

2. **Registro de solicitações:** Nos dez primeiros dias de um determinado mês, os usuários-chave de cada sistema devem formalizar as manutenções de software definidas na etapa anterior, através do registro de solicitações de manutenção, por exemplo, em um sistema de controle de solicitações de mudanças.

3. **Elaboração de propostas de projetos:** Após o registro das solicitações, os analistas de sistemas tem aproximadamente uma semana para elaborar propostas de projetos de manutenção - em conjunto com a equipe de desenvolvimento - para serem apreciados pelos usuários-chave. As estimativas iniciais de esforço e prazo previstos para conclusão das solicitações devem ser mensuradas com base nos dados históricos, considerando as tecnologias envolvidas, experiência e disponibilidade da equipe e a complexidade e tamanho da modificação.

4. **Avaliação e priorização de projetos:** Ao final da terceira semana de cada mês, uma comissão de priorização de projetos, formada pelos líderes da equipe de manutenção e usuários-chave, deve se reunir para avaliar e priorizar as propostas de projetos. O painel de projetos deve apresentar a situação atualizada dos projetos em execução. A priorização deve se basear nas informações desse painel e na capacidade produtiva da equipe. As propostas aprovadas serão convertidas em projetos e, para cada projeto, será iniciado um ciclo de execução distinto, baseado na metodologia de desenvolvimento adotada pela instituição.

5. **Especificação de requisitos:** Ao término da aprovação do escopo do projeto, inicia-se a especificação detalhada das necessidades dos usuários, envolvendo a equipe de manutenção e os principais usuários do sistema em questão.

6. **Validação das especificações de requisitos:** Os artefatos construídos na etapa anterior são encaminhados para a área de qualidade para verificação da conformidade com os padrões definidos pelo SEPG (Software Engineering Process Group) [Fowler et al. 1990] da instituição. Os profissionais da área de garantia da qualidade produzem um relatório de avaliação com o resultado da verificação dos artefatos e, caso estes estejam aderentes aos propósitos dos usuários, o projeto de manutenção é então encaminhado para a implementação.

7. **Implementação:** Após receber o projeto, os desenvolvedores analisam os artefatos produzidos até então, planejam a execução desta atividade e constroem o desenho para implementação do código e a definição dos testes unitários.

8. **Validação da implementação:** Após concluir as atividades de implementação do projeto de manutenção em questão e realizar os testes funcionais de primeiro nível, a equipe de desenvolvedores libera uma nova versão do software no ambiente de homologação para que esta seja verificada pelo arquiteto de software e pelos analistas de sistemas.

9. **Testes:** Os projetos de manutenção aprovados na validação funcional são encaminhados para a equipe de garantida da qualidade. Nesta etapa são realizados os procedimentos de testes definidos para o projeto. As atividades de planejamento dos testes devem ocorrer em paralelo à implementação do projeto.

10. **Implantação:** Após concluídas as atividades de testes, a implantação dos projetos aprovados é realizada em ambiente de produção, de acordo com os prazos acordados com os usuários.

Mostra-se na Figura 1 o fluxo das etapas do Processo para Agrupamento de Solicitações de Manutenções de Software. No PASM, as demandas de manutenção de software que necessitam de atendimento urgente – por exemplo, manutenções corretivas ou adaptativas com datas críticas para atendimento – devem ser reportadas a um suporte de informática de primeiro nível, que segue um procedimento específico para caracterização e triagem de solicitações urgentes – definido pela instituição – e as encaminha à área responsável pelo atendimento. As solicitações urgentes , sempre que possível, são planejadas para execução em um dia pré-determinado, por exemplo às segundas-feiras. Nesse

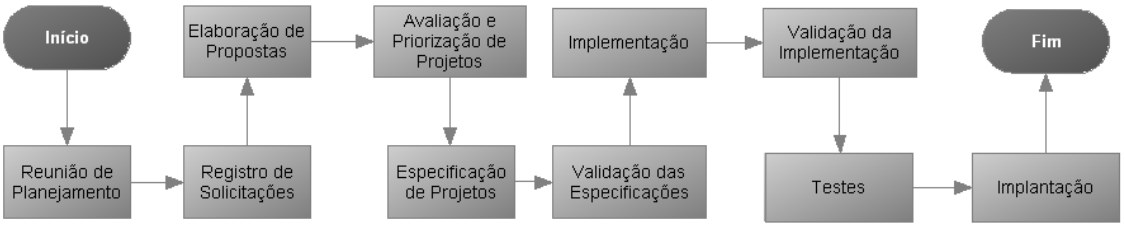

Figura 1. Fluxo de etapas do PASM.

dia a equipe reserva um número de horas específico de tal forma que não prejudique a realização dos projetos priorizados e em andamento. Para as ocasiões que apresentarem a necessidade de se estender o atendimento para além do dia pré-determinado, deve-se negociar a suspensão ou não do atendimento junto aos usuários. No caso da ocorrência do registro de uma solicitação urgente cujo atendimento não pode aguardar até o próximo plantão, a necessidade deve ser informada durante o registro da solicitação para que a gerência da equipe de manutenção possa avaliar a veracidade da requisição e caso julgue pertinente deverá delegar de imediato o atendimento da solicitação a um membro da equipe.

A Figura 2 apresenta uma instância do calendário de atividades do modelo de agrupamento e execução de projetos de manutenção de software.

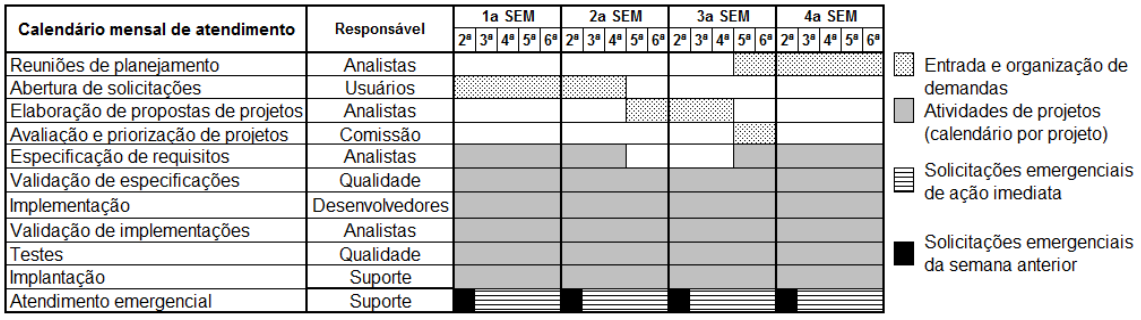

Figura 2. Calendário mensal de atendimento de demandas de manutenção de software.

3. Estudo de Caso

O Processo de Agrupamento de Solicitações de Manutenção de Software proposto neste trabalho está sendo utilizado como ferramenta de apoio ao processo de manutenção de software do DATAPUC, a divisão de tecnologia da PUC Minas. O DATAPUC é responsável pelo desenvolvimento e manutenção dos sistemas utilizados nos diversos departamentos da PUC Minas e seu portfólio é composto por 40 sistemas que possuem aproximadamente 4 milhões de linhas de código. Atualmente, o DATAPUC possui 34 profissionais responsáveis pelo desenvolvimento e análise de sistemas, garantia da qualidade, gestão de configuração e gerenciamento de projetos.

A implantação do PASM se efetivou em novembro de 2008 e, a partir deste momento, as diretrizes que constituem esse processo passaram a ser aplicadas no tratamento de todas as solicitações de manutenção de software registradas pelos usuários.

O presente estudo apresenta e discute os benefícios proporcionados pela implantação do PASM. Para isso, foram coletados e analisados diversos dados sobre as manutenções realizadas pelo DATAPUC antes e após a implantação do processo.

O restante desta seção está organizado conforme descrito a seguir. Inicialmente, a Seção 3.1 descreve a metodologia adotada para avaliação dos benefícios gerados pelo PASM. A Seção 3.2 descreve a amostra de dados adotada nesta avaliação. Por fim, a Seção 3.3 apresenta e discute os resultados obtidos.

3.1. Metodologia

As demandas de manutenção registradas durante os meses de fevereiro a outubro de dois anos distintos e consecutivos, 2008 e 2009, foram extraídas da base de dados do sistema de acompanhamento de solicitações de manutenções do DATAPUC. Também foi extraído desse sistema o respectivo conjunto de lançamentos de horas trabalhadas por cada funcionário envolvido nas atividades realizadas durante o atendimento das solicitações.

Para facilitar a identificação e interpretação dos benefícios obtidos com a implantação do PASM, três tipos principais de dados foram coletados:

1. **Número de Solicitações**: As solicitações foram classificadas em uma estrutura hierárquica, na qual os níveis são representados pelos atributos `Ano`, `Mês`, `Sistema` e `Tipo de Solicitação`. A classificação proposta permite identificar tendências de crescimento ou declínio nas variáveis mensuradas em cada período e para cada tipo de solicitação, já que o domínio do atributo `Tipo de Solicitação` é composto pelos valores `Projeto` e `Urgente`. Essa separação é importante, visto que essas solicitações são tratadas de forma distinta no fluxo de trabalho proposto pelo PASM. A estrutura foi utilizada para mensurar o número absoluto de solicitações registradas pelos usuários e o número de solicitações concluídas ao longo dos meses considerados no estudo.

2. **Quantidade de Horas Trabalhadas**: Os lançamentos de horas trabalhadas foram classificados por `Ano`, `Mês`, `Tipo de Solicitação`, `Funcionário` e `Atividade`. Essa classificação permite identificar as solicitações nas quais os funcionários trabalharam e as solicitações cuja execução se desdobrou por mais de um mês.

3. **Estimativa do Tamanho das Solicitações de Manutenção**: Como avaliação complementar, o tamanho da alteração realizada no software após a execução de cada solicitação foi identificado e comparado com as demais variáveis mensuradas. Porém, os dados das solicitações de manutenção existentes não contemplavam uma métrica que permitisse determinar o tamanho das alterações. Métricas como Pontos de Função ou Pontos de Caso de Uso são amplamente utilizadas na estimativa de custo e esforço de projetos de desenvolvimento de software. Contudo, para uma estimativa concisa é preciso uma avaliação pontual de cada solicitação e esta avaliação requer conhecimento detalhado do comportamento e da estrutura do software a ser modificado. Esses aspectos inviabilizam o uso dessas métricas em um volume considerável de solicitações que já foram concluídas. Dessa forma, para viabilizar a produção de evidências sobre a evolução da produ-

ção dos funcionários ao longo do tempo, foi adotada como unidade de tamanho o número de linhas de código modificadas a cada manutenção.

Com o objetivo de reproduzir a seqüência das alterações realizadas no código fonte das aplicações e calcular o número de linhas de código modificadas em cada manutenção foram executados dois procedimentos para extrair e determinar quais linhas de código foram alteradas, excluídas ou inseridas em cada versão dos arquivos. Esses procedimentos são detalhados a seguir.

Procedimentos para Cálculo do Número de Linhas Modificadas: o primeiro procedimento incluiu o desenvolvimento de um utilitário que percorre em ordem cronológica o histórico de versões dos arquivos presentes no repositório de versões utilizado pelo DATAPUC (Microsoft Visual Source Safe[1]). Para cada arquivo encontrado nessa pesquisa, foi feita uma comparação textual entre a versão anterior e a versão modificada. O processo de extração foi executado separadamente para cada sistema considerado no estudo, percorrendo todos os *check-ins* enviados ao repositório de versões. Os novos arquivos presentes nos *check-ins* considerados foram comparados com um arquivo vazio. A comparação das versões dos arquivos modificados foi realizada por meio da ferramenta Compare It[2]. Essa ferramenta produz um relatório contendo o número de linhas alteradas, excluídas e inseridas em cada arquivo. Os sistemas analisados foram implementados nas linguagens Delphi e Java e possuem objetos de banco de dados desenvolvidos em Transact SQL. Apenas arquivos em formato texto foram considerados, incluindo as extensões .pas, .dfm, .nfm, .cfg, .java, .sql, .udf, .prc, .trg, .viw, .xml e .cs.

No segundo procedimento, os resultados da comparação dos arquivos foram submetidos a um segundo utilitário construído para processar os relatórios e extrair o número de linhas alteradas, excluídas e inseridas. Esses valores foram armazenados em um banco de dados relacional para facilitar as análises posteriores.

Por fim, duas estratégias foram usadas para vincular os *check- ins* registrados na base de dados relacional com suas respectivas solicitações de manutenção:

1. Durante a realização do *check-in*, algumas equipes têm como padrão descrever em qual solicitação estavam trabalhando no momento do envio do *check-in* para o repositório de versões. Um *parser* foi então construído para identificar quais comentários registrados continham o código da solicitação trabalhada. No caso dessa identificação ser bem sucedida, o vínculo entre esses *check-ins* e as solicitações identificadas foi estabelecido diretamente.

2. A segunda estratégia é aplicada apenas sobre os registros não vinculados pela primeira estratégia. Essa estratégia vincula os *check-ins* com as solicitações quando o funcionário responsável pelo *check-in* contém um registro de lançamento de horas cujo período abrange a data de envio do *check-in*. O vínculo com a solicitação somente foi estabelecido caso a solicitação de manutenção identificada seja referente ao mesmo sistema do *check-in*.

[1] `http://msdn.microsoft.com/en-us/library/ms181038(VS.80).aspx`.
[2] `http://www.grigsoft.com`.

Nas duas estratégias utilizadas foram consideradas apenas as solicitações em que ocorreu pelo menos uma vez a atividade de programação, eliminando as solicitações de análise de dados e geração de relatórios, solicitações canceladas ou solicitações não aprovadas pelos usuários-chave.

3.2. Caracterização da Amostra

Para realização deste trabalho foram avaliados treze sistemas do DATAPUC. A escolha desses sistemas foi determinada pelo percentual de solicitações registradas, pelo total de horas trabalhadas no atendimento das solicitações e pelo valor estratégico dos sistemas. A Tabela 1 apresenta a distribuição das solicitações entre os treze sistemas avaliados. Dentre esses sistemas, o GRA, SAL e SGA são os três principais sistemas mantidos pelo DATAPUC e, portanto, apresentam o maior número de solicitações registradas.

Tabela 1. Total de solicitações por sistema.

Sigla	Descrição	Solicitações
GAC	Sistema de Gestão de Atendimento de Clínicas	54
GPE	Sistema de Gestão de Pesquisa e Extensão	140
GPI	Sistema de Gestão de Processos de Inscrição	75
GPJ	Sistema de Gestão de Práticas Jurídicas	144
GPP	Sistema de Gestão de Projetos Pedagógicos	73
GPS	Sistema de Gestão de Processos Seletivos	172
GRA	Sistema de Gestão de Receita Acadêmica	725
SAE	Sistema de Autorização de Pagamento Eletrônico	112
SAL	Sistema Acadêmico de Pós-Graduação *Lato Sensu*	530
SGA	Sistema de Gestão Acadêmica	1.052
SGC	Sistema de Gestão de Colégios	346
SGE	Sistema de Gestão de Estágios	139
SOL	Sistema de Solicitações	69
		3.631

A Tabela 2 mostra que as solicitações dos treze sistemas destacados para análise representam 84,91% do total de solicitações registradas pelos usuários do DATAPUC no período deste estudo. Além disso, essa tabela apresenta a quantidade de solicitações que foram efetivamente executadas, visto que as solicitações canceladas ou não aprovadas durante as reuniões de priorização com os usuários foram desconsideradas. Com o propósito de analisar a produtividade dos desenvolvedores, as comparações entre linhas de código modificadas e horas trabalhadas consideraram apenas as solicitações cujo lançamento de horas incluiu pelo menos uma atividade de programação. Ao considerarmos as solicitações com atividade de programação, as solicitações dos sistemas avaliados abrangem 86,37% do total desse subconjunto de solicitações.

A Tabela 3 apresenta o número de horas destinadas pelos desenvolvedores do DATAPUC para implementar as solicitações consideradas no estudo. O esforço total mobilizado para atendimento das solicitações registradas para os treze sistemas considerados representa 86,99% das horas trabalhadas (isto é, 21.475 horas de um total de 24.686 horas) e 87,05% das horas trabalhadas em solicitações com atividade de programação.

Tabela 2. Total de solicitações.

	Todos os Sistemas	Sistemas Avaliados	%
Solicitações registradas	4.276	3.631	84,91
Solicitações executadas	2.968	2.548	85,85
Solicitações com programação	2.927	2.528	86,37

Tabela 3. Total de horas trabalhadas.

Atividade	Portfólio Geral			Sistemas Avaliados		
	Total	2008	2009	Total	2008	2009
Todas	24.686	12.285	12.401	21.475	10.597	10.878
Programação	13.333	7.515	5.818	11.607	6.473	5.134

3.3. Resultados

A análise conduzida neste trabalho avaliou os impactos da implantação do PASM em diversos aspectos relacionados à priorização e tratamento das demandas de manutenção de software. Entre os fatores investigados encontram-se o número absoluto de solicitações registradas, o volume de horas trabalhadas em cada solicitação e a produtividade dos desenvolvedores no atendimento das solicitações urgentes.

Por meio da análise do número de solicitações registradas é possível avaliar a variação da demanda de serviços de manutenção de software. O volume de horas trabalhadas complementa a análise do número absoluto de solicitações registradas, visto que permite determinar como cada tipo de solicitação é tratado pela equipe de manutenção e demonstra o real impacto da priorização dos projetos no custo da manutenção dos sistemas. O impacto do PASM na produtividade dos desenvolvedores é discutido com base nos dados coletados sobre o volume de linhas de código modificadas em cada manutenção.

Nas subseções seguintes, cada um dos indicadores mencionados será analisado. O objetivo principal é comparar os valores desses indicadores após a implantação do PASM com seus valores antes da implantação do processo.

3.3.1. Número de Solicitações

A Tabela 4 mostra que foram registradas 1.684 solicitações urgentes em 2008 e 1.648 solicitações urgentes em 2009. Logo, o número de solicitações urgentes registradas após a implantação do PASM diminuiu apenas 2,18%.

Tabela 4. Quantidade de solicitações registradas.

Solicitações registradas								
Sistema	Urgentes				Projetos			
	2008		2009		2008		2009	
	Qtd.	%	Qtd.	%	Qtd.	%	Qtd.	%
Total	1.684	50,54	1.648	49,46	42	35,00	78	65,00
Média Mensal	129,54	49,06	126,77	50,94	3,23	22,84	6,00	61,77

A pequena queda no número de solicitações urgentes pode ser explicada com base na maneira como esse tipo de solicitação passou a ser tratado no novo fluxo de trabalho. No PASM, sempre que possível, as solicitações urgentes são realizadas em dias pré-determinados, por exemplo, em uma segunda-feira. Neste dia, os desenvolvedores realizam um plantão para atendimento das manutenções que devem ser executadas em um período menor que o tempo médio para execução de um projeto de manutenção e, dessa forma, uma parcela fixa da capacidade de produção da equipe é destinada ao atendimento de solicitações urgentes. Como a resposta ao atendimento desse tipo de solicitação é mais rápida do que a realização de um projeto de manutenção, alguns usuários-chaves chegam a registrar várias pequenas solicitações urgentes ao invés de elaborar um projeto de manutenção. Isso indica que a utilização de um plantão para atendimento de solicitações urgentes pode comprometer as premissas essenciais do PASM se a avaliação da natureza das solicitações de manutenção não for criteriosamente avaliada.

Por fim, a Tabela 4 revela um aumento expressivo de 85,71% no número de solicitações de manutenção que foram tratadas como projeto (42 projetos de manutenção realizados em 2008; e 78 projetos realizados em 2009, após a implantação do PASM). Este aumento mostra que um dos principais objetivos do PASM foi cumprido: fomentar o agrupamento de solicitações de manutenção em projetos de manutenção de software.

3.3.2. Número de Horas Trabalhadas

Em relação à quantidade de horas trabalhadas, a Tabela 5 mostra que a diferença entre o total de horas envolvendo todas as solicitações de 2008 e de 2009 não apresentou variação significativa. De acordo com essa tabela, em 2009 houve um aumento de horas trabalhadas de apenas 0,90% (10.592 horas em 2008; e 10.878 horas em 2009).

Tabela 5. Comparação entre horas trabalhadas em todas as solicitações de manutenção e em solicitações de manutenção com atividades de programação.

Horas trabalhadas								
Sistema	Geral				Programação			
	2008		2009		2008		2009	
	Qtd.	%	Qtd.	%	Qtd.	%	Qtd.	%
Total	10.592	49,33	10.878	50,67	6.468	55,75	5.135	44,25
Média	814,71	49,31	836,72	50,69	497,49	44,29	394,94	40,33

Esses dados autorizam uma comparação absoluta entre o número de horas dedicadas pelas equipes de manutenção do DATAPUC em 2008 e em 2009 nas diversas atividades como análise, programação, testes, gerência de qualidade, etc. Essa comparação é mostrada na Figura 3. Nessa figura, verifica-se que após a implantação do PASM aumentou o número de horas dedicadas a três atividades: análise, implantação e qualidade. Por outro lado, diminuiu o número de horas dedicadas à programação.

A conclusão derivada de uma análise da Figura 3 é imediata. Ao favorecer a criação de projetos de manutenção (Tabela 4), o PASM propiciou que as equipes dedicassem mais tempo às atividades de análise e qualidade e menos tempo a atividades de codificação. Em resumo, as manutenções no PASM incluem atividades que reconhecidamente

agregam valor à qualidade de um software e ajudam a antecipar a identificação de defeitos e a mitigação dos riscos dos projetos.

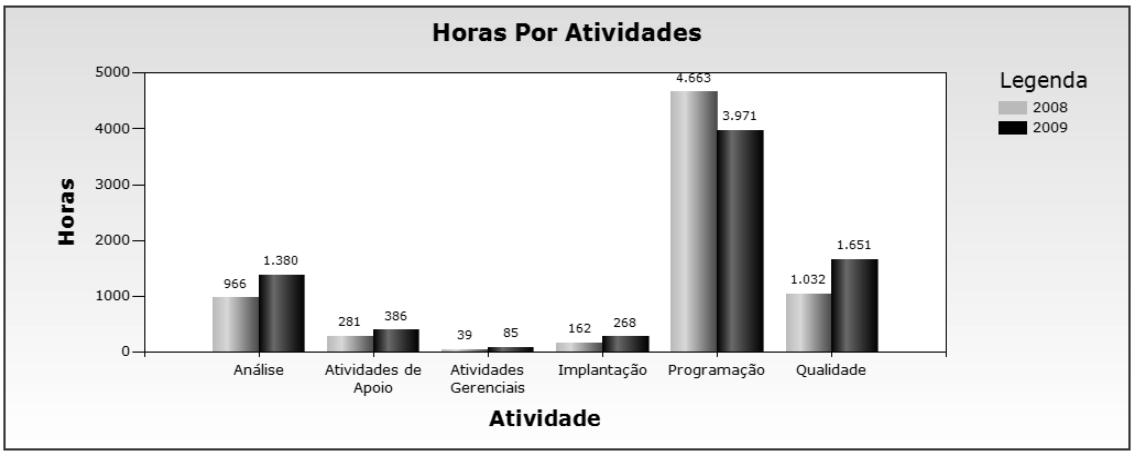

Figura 3. Horas trabalhadas por atividade.

3.3.3. Duração das Manutenções

A Figura 4 mostra que após a implantação do PASM ocorreu uma queda no tempo médio para implementação de manutenções urgentes. Por exemplo, em 2009 a média de horas trabalhadas permaneceu inferior a 2008 em praticamente todos os meses.

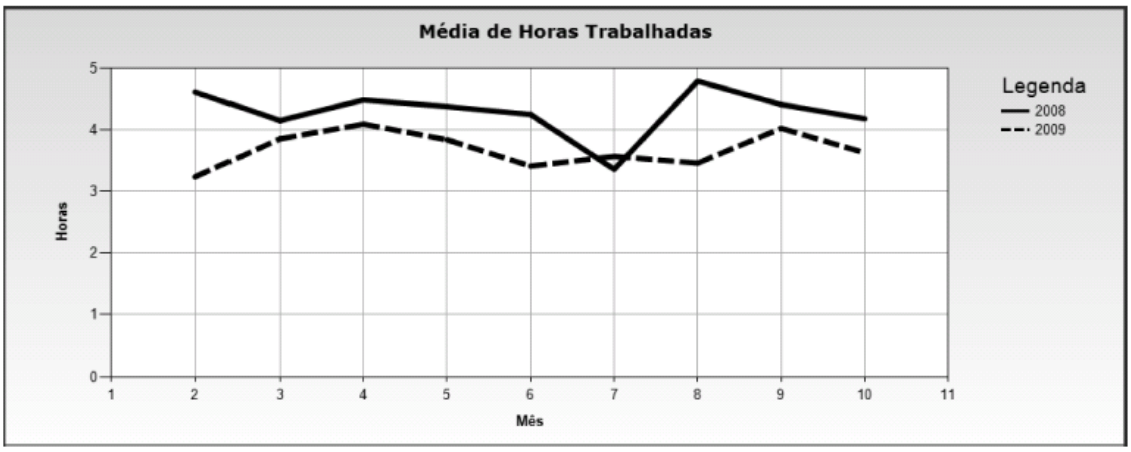

Figura 4. Média de horas trabalhadas em solicitações urgentes.

Já a Figura 5 mostra que em 2009 o total de horas trabalhadas em solicitações urgentes diminuiu 27,73%, ou 1.786,07 horas. Em média, em 2009, os desenvolvedores do DATAPUC trabalharam 338,97 horas em atividades de manutenção urgente. Em 2008, esse número foi um pouco maior, 357,67 horas. Os valores foram obtidos através da divisão entre o total de horas trabalhadas e o total de solicitações atendidas. Conforme relatado anteriormente, o número de solicitações urgentes registradas não apresentou uma queda expressiva em 2009 e o total de horas trabalhadas com solicitações urgentes caiu. Com isso, em 2009, o tempo médio de atendimento das solicitações urgentes foi inferior.

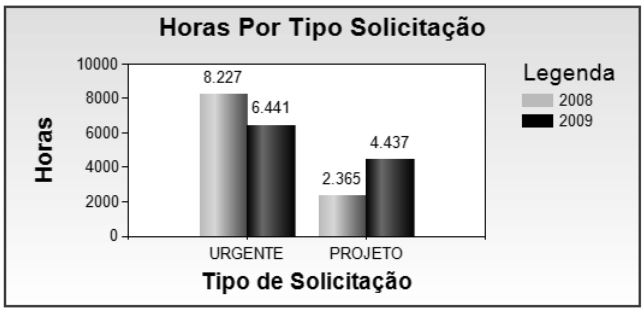

Figura 5. Horas trabalhadas por tipo de solicitação.

Com as tentativas de enquadrar as solicitações de manutenção adaptativas e evolutivas sempre em projetos de manutenção, as solicitações urgentes tornaram-se mais objetivas e tiveram o escopo reduzido, envolvendo alterações mais pontuais, focadas principalmente na correção de erros.

3.3.4. Produtividade dos Programadores

A Tabela 6 mostra a razão entre a quantidade de linhas de código modificadas (alteradas, excluídas e inseridas) pela quantidade de horas trabalhadas com programação. Pode-se observar que em 2009 (após a implantação do PASM), os desenvolvedores modificaram em média 610 linhas por hora de trabalho de programação. Em 2008 (antes da implantação do PASM), esse número foi bem menor (342 linhas modificadas por hora). Os números mostrados incluem não apenas linhas manualmente modificadas, mas também linhas de código geradas automaticamente pelas ferramentas de programação usadas pelo DATAPUC (geradores de relatórios, editores de interface gráfica, etc). A produtividade dos profissionais envolvidos na implementação das manutenções também é influenciada pela estabilidade obtida pelos sistemas na medida em que seu tempo de vida aumenta e pelo conhecimento adquirido pelos desenvolvedores. Apesar do perfil da equipe não ter sofrido mudanças significativas, o nível de conhecimento dos desenvolvedores sobre o código fonte poder também ter contribuído para o aumento da produtividade no atendimento das solicitações de manutenção [Tan and Mookerjee 2005].

Tabela 6. Média de linhas modificadas por hora trabalhada nas atividades de programação durante o atendimento de solicitações de manutenção de software.

Ano	Linhas Modificadas por Hora
2008	342
2009	610

3.3.5. Resumo dos Resultados

Nesta seção, apresenta-se um resumo e uma avaliação geral dos diversos resultados descritos nas seções anteriores.

- O número absoluto de solicitações de manutenção urgentes antes e após a implantação do PASM não sofreu uma mudança considerável (permanecendo

pouco superior a 1.600 solicitações por ano).

- No entanto, as manutenções urgentes tratadas após a implantação do PASM apresentaram duas características positivas: (a) foram implementadas em menos tempo (via de regra, sempre em menos de quatro horas; antes do PASM esse tempo era quase sempre superior a quatro horas); (b) mesmo sendo implementadas em menos tempo, implicaram na modificação de um número maior de linhas de código. Essas duas características evidenciam que houve um ganho de escala importante após a implantação do PASM, isto é, os desenvolvedores trabalharam menos tempo, mas produziram mais linhas de código.

- O número absoluto de manutenções tratadas como projetos de manutenção aumentou de forma considerável (de 42 projetos de manutenção antes do PASM, para 78 projetos após a implantação do processo). Como resultado, as equipes de manutenção passaram a dedicar mais tempo a atividades de especificação de requisitos, análise e controle de qualidade e menos tempo a atividades de programação. Esse fato é um indicativo importante de que houve um aumento na qualidade dos sistemas mantidos pelo DATAPUC após a implantação do PASM.

4. Trabalhos Relacionados

Os estudos relacionados a manutenção de software podem ser divididos em dois grupos. O primeiro abrange as pesquisas aplicadas ao desenvolvimento de modelos para estimar e otimizar os custos de manutenção. O segundo grupo envolve os trabalhos que almejam entender e evoluir os processos e métodos para guiar a fase de manutenção de software.

Modelos de Estimativa de Custos: Diversos estudos têm investigado modelos para estimativa de custos para realização de manutenção de software. Esses modelos utilizam técnicas distintas e avaliam as variações proporcionadas por diversos fatores. Banker e Slaughter investigaram os benefícios oferecidos pelo tratamento de manutenção como projetos de software [Banker and Slaughter 1997]. Eles avaliaram 129 projetos de manutenção de software de uma grande organização financeira, empregando o modelo não paramétrico chamado DEA (*Data Envelopment Analysis*). O estudo apontou uma redução de 36% proporcionada pelo agrupamento das solicitações em projetos de manutenção.

Tan e Mookerjee propuseram um modelo para apoiar o controle proativo de solicitações de manutenção de software [Tan and Mookerjee 2005]. Este modelo inclui não apenas informações para tomada de decisões sobre manutenções como também indicadores sobre a necessidade de substituição dos sistemas quando o custo de manutenção inviabiliza a evolução do produto de software. Embora possibilite simular características como nível de degradação do software, curva de aprendizagem da equipe, entre outros fatores, este modelo matemático não foi avaliado em projetos reais e a estimativa precisa de parte dos parâmetros utilizados é uma tarefa árdua.

Feng, Mookerjee e Sethi apresentaram uma solução baseada em programação dinâmica para o problema de manutenção de software [Feng et al. 2006]. Assumindo que a taxa de chegada das solicitações de manutenção é aderente à distribuição de Poison, o modelo objetiva otimizar o custo total para a manutenção de sistemas.

Processos e Métodos: MANTEMA é uma metodologia para gerenciamento do processo de manutenção de software [Polo et al. 2002]. Esta metodologia propõe um conjunto de atividades e tarefas iterativas para manipular as solicitações de manutenção. Contudo, esta metodologia não inclui uma atividade explícita onde as solicitações são enfileiradas, avaliadas e então agrupadas em projetos.

O modelo SMLC (*Software Maintenance Life Cycle Model*) distribui o tempo de vida de um software em quatro estágios não estacionários (introdução, crescimento, amadurecimento e declínio) [Kung and Hsu 1998]. Nos três primeiros estágios, os tipos de manutenção predominantes são respectivamente suporte ao usuário, correções e evoluções. No último estágio, usuários e gerentes de projeto iniciam o planejamento da substituição do sistema. Posteriormente, Kung propôs um método de decisão quantitativo para determinar as mudanças de um estágio para o próximo. O objetivo é apoiar os gerentes na previsão de demandas de manutenção de software [Kung 2004].

Niessink e Van Vliet investigaram as diferenças entre a manutenção e o desenvolvimento de software [Niessink and van Vliet 2000]. O argumento central deste trabalho é que a manutenção deve ser tratada como fornecimento de serviço, enquanto o desenvolvimento está preocupado com a construção de produtos. Dessa forma, a manutenção deve ser orientada por processos diferentes dos processos utilizados durante o desenvolvimento. Com base nessas conclusões, o estudo propõe um modelo de maturidade focado na capacidade de fornecimento de serviços de tecnologia da informação.

5. Conclusões

O crescimento do uso de sistemas em diversos setores da sociedade, o aumento da complexidade dos domínios nos quais esses são aplicados e o aumento do tempo médio de vida dos sistemas legados demandam manutenções eficientes e eficazes e, consequentemente, motivam o desenvolvimento de processos que apóiem o planejamento de suas atividades. Este artigo apresentou o Processo de Agrupamento de Solicitações de Manutenção (PASM) que objetiva organizar as solicitações de manutenção de software em projetos, auxiliando o planejamento e a gestão do atendimento das solicitações e dessa forma reduzindo a ocorrência de solicitações urgentes.

O PASM é executado em um ciclo mensal de elaboração, avaliação e aprovação de propostas de projetos de manutenção de software. Cada ciclo do processo é constituído por dez etapas realizadas ao longo de quatro semanas. Este processo foi aplicado no DATAPUC, Divisão de Tecnologia da PUC Minas ao longo do ano de 2009. A partir da análise e comparação das características das solicitações de manutenção dos sistemas mantidos pelo DATAPUC percebeu-se um aumento de 18,76% nas horas trabalhadas em projetos de manutenção e um declínio de 27,73% nas horas trabalhadas em solicitações urgentes. Esses números apontam para a melhoria dos produtos de software do DATAPUC e a redução dos custos operacionais das atividades de manutenção devido aos ganhos de escala proporcionados pelos projetos de manutenção.

Como trabalho futuro, pretende-se formalizar a definição do PASM, usando para isso um meta-modelo para definição de processos de software, como o SPEM (*Software Process Engineering Metamodel*) [Object Management Group 2002]. Basicamente, o SPEM define diversos estereótipos UML para a modelagem de processos de software. Pretende-se também realizar uma avaliação mais aprofundada dos benefícios proporci-

onados pelo PASM no DATAPUC, possivelmente considerando um horizonte de tempo maior do que nove meses. Por fim, pretende-se incentivar o uso do PASM em outras organizações de desenvolvimento de software, de forma a obter novos estudos de caso.

Agradecimentos: Este trabalho foi apoiado pelo CNPq e pela FAPEMIG.

Referências

Banker, R. D. and Slaughter, S. A. (1997). A field study of scale economies in software maintenance. *Management Science*, 43(12):1709–1725.

Erlikh, L. (2000). Leveraging legacy system dollars for e-business. In *IEEE IT Pro*, pages 17–23.

Feng, Q., Mookerjee, V. S., and Sethi, S. P. (2006). Optimal policies for the sizing and timing of software maintenance projects. *European Journal of Operational Research*, 173(3):1047–1066.

Fowler, P., Rifkin, S., and Card, D. M. (1990). Software engineering process group guide. Technical report, Software Engineering Institute. `http://www.sei.cmu.edu/reports/90tr024.pdf`.

Kung, H.-J. (2004). Quantitative method to determine software maintenance life cycle. In *20th IEEE International Conference on Software Maintenance (ICSM)*, pages 232–241.

Kung, H.-J. and Hsu, C. (1998). Software maintenance life cycle model. In *14th IEEE International Conference on Software Maintenance (ICSM)*, pages 113–121.

Malta, M. N., de Almeida, H. M., Valente, M. T., Pietrobon, C. A., and Marques-Neto, H. T. (2009). Modelo de gestão de demandas de manutenção de software: a experiência da PUC Minas. In *VI Workshop de Manutenção de Software Moderna (WMSWM)*, pages 1–4.

Niessink, F. and van Vliet, H. (2000). Software maintenance from a service perspective. *Journal of Software Maintenance and Evolution: Research and Practice*, 12(2):103–120.

Object Management Group (2002). Software process engineering meta-model.

Polo, M., Piattini, M., and Ruiz, F. (2002). Using a qualitative research method for building a software maintenance methodology. *Software Practice and Experience*, 32(13):1239–1260.

Sommerville, I. (2004). *Software Engineering*. Addison Wesley, 7th edition edition.

Sutherland, J. (2000). Business objects in corporate information systems. *ACM Computing Surveys*, 27(2):274–276.

Tan, Y. and Mookerjee, V. S. (2005). Comparing uniform and flexible policies for software maintenance and replacement. *IEEE Transactions on Software Engineering*, 31(3):238–255.