Melhorias no Processo de Manutenção de Software Colaborativa do Laboratório de Engenharia de Software/UFMS

Maria Istela Cagnin[1], Geraldo Barbosa Landre, Leandro Magalhães de Oliveira, Marcelo de Andréa Nahabedian, Andre Hora, Débora Maria Barroso Paiva[2]

Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Caixa Postal 549 – 79070-900 – Campo Grande – MS – Brasil

{istela, geraldo, debora}@facom.ufms.br, {andrehoraa, leandro.oliveira.ms}@gmail.com, joshua@personalhelp.com.br

Resumo. *Durante a definição e melhoria de um processo de software adequado para uma organização é necessário considerar, além de métodos e práticas de Engenharia de Software, o conhecimento e as habilidades dos membros da equipe de desenvolvimento. Isso foi constatado durante a melhoria de um processo colaborativo de manutenção, denominado ProFap, utilizado no Laboratório de Engenharia de Software (LEDES) da UFMS. Este artigo apresenta as deficiências observadas no ProFap que motivaram a sua melhoria e evolução para o processo ProLedes. Os papéis, as atividades, os artefatos e a personalização de ferramentas computacionais para apoiar o ProLedes são apresentados. Com o intuito de comprovar os resultados obtidos com o ProLedes, diversas evidências são apresentadas.*

Abstract. *During the software process definition and improvement for an organisation, it is fundamental to consider, in addition to Software Engineering methods and practices, the development team knowledge and skills. This was detected during the improvement of a maintenance collaborative process, namely ProFap, which is used in the Software Engineering Lab (LEDES) at UFMS. This paper describes the shortcomings we noted in ProFap that motivated its improvement and evolution to the ProLedes process. We present the roles, activities, artifacts and tool customization to support ProLedes. In order to verify ProLedes results, we provide several evidences.*

1. Introdução

Com a expansão do uso de sistemas de software nas diferentes áreas do conhecimento nota-se o aumento da competitividade entre as organizações de software para atender as demandas. Com isso, torna-se evidente a busca dessas organizações por diferenciais como forma de se manter e crescer nesse ambiente, em que a qualidade dos produtos e a gerência de recursos tornam-se indispensáveis.

[1] Apoio financeiro da Fundect - T.O. nº 115/2014 - SIAFEM nº 23710.
[2] Apoio financeiro da Fundect - T.O. nº 219/2014 - SIAFEM nº 23963.

De acordo com a literatura, investimentos na qualidade de processos de desenvolvimento de software podem melhorar a qualidade dos produtos disponibilizados no mercado, aumentar a produtividade da equipe de desenvolvimento e reduzir custos (FUGGETA, 2000; PRESSMAN, 2016).

De acordo com Viana *et al*. (2014), uma iniciativa de melhoria de processo de software pode ser executada de acordo com as necessidades organizacionais ou pode seguir boas práticas de Engenharia de Software promovidas pelos modelos de maturidade em melhoria de processo de software, como o CMMI-DEV (*Capacity Maturity Model Integration for Development*) (SEI, 2010). Assim, independente dos mecanismos utilizados para apoiar a melhoria de processo de software, a adoção de processos de desenvolvimento alinhados aos objetivos estratégicos da organização tendem a gerar bons resultados.

A equipe de professores e colaboradores do Laboratório de Engenharia de Software (LEDES) da Faculdade de Computação (Facom) da Universidade Federal de Mato Grosso do Sul (UFMS) definiram um processo colaborativo de desenvolvimento e manutenção de software, denominado ProFap (CAGNIN *et al*., 2013), para atender as demandas de um Sistema de Informação de Gestão de Fundações de Amparo à Pesquisa – SIGFAP (Carromeu *et al*., 2010; Turine *et al*., 2011). Esse sistema tem como intuito gerenciar os projetos de pesquisa e facilitar o trâmite desde a submissão da proposta até a fase de prestação de contas. Para unir esforços para a manutenção e utilização de uma única versão do SIGFAP para a gestão das FAPs foi criada a Rede SIGFAP junto ao Conselho Nacional das Fundações de Amparo à Pesquisa (CONFAP).

À medida que o ProFap passou a ser utilizado pela equipe do LEDES diversos problemas foram detectados e contribuíram indiretamente para diversas iniciativas de melhorias no processo. Essas iniciativas de melhorias estavam alinhadas ao planejamento estratégico do Ledes, realizado em 2015, em que foi vislumbrada a importância de melhorar o processo ProFap. A inclusão desse item no planejamento estratégico levou a atribuição de tarefas a alguns membros (gerente de projeto e três professores da área de Engenharia de Software) com o intuito de começar a estudar as melhores práticas que poderiam ser adotadas e ferramentas de apoio. O resultado obtido foi o processo ProLedes, apresentado neste artigo.

Assim, o objetivo deste artigo é descrever os problemas existentes com o uso do ProFap, tais como as dificuldades de utilizá-lo na prática, a existência de papéis sem responsabilidades bem definidas, a ausência de atividades de análise, estimativa e planejamento das tarefas e testes e a ausência do histórico das comunicações interna e externa. Além disso, neste artigo são descritas as iniciativas de melhorias propostas para dirimir esses problemas. Como consequência, é apresentado o processo ProLedes, definido a partir da evolução do ProFap, com o detalhamento de suas atividades, papéis, artefatos e ferramentas computacionais de apoio. Para destacar as vantagens obtidas com a implantação desse novo processo são também apresentadas evidências das melhorias realizadas. Assim, ressalta-se que a principal contribuição deste artigo é relatar uma experiência na melhoria de processo no contexto de instituições públicas e laboratórios de pesquisa. Outras iniciativas de melhorias de processos de software em

ambiente acadêmico podem ser encontradas na literatura, como a de MENDES *et al*. (2011) e a de SANTOS *et al*. (2015).

A escrita deste artigo está organizada em mais cinco seções. Na Seção 2 é apresentado o processo ProFap e são discutidos os problemas encontrados após a sua implantação. Na Seção 3 é apresentado o processo ProLedes. Na Seção 4 são apresentadas as evidências das melhorias obtidas com a implantação do ProLedes. Por fim, na Seção 5 são apresentadas as contribuições, lições aprendidas e sugestões de trabalhos futuros.

2. ProFap

O ProFap (CAGNIN *et al*., 2013) é um processo colaborativo de desenvolvimento e manutenção com integração de ferramentas de apoio a diversas atividades, tais como o Redmine (para gerenciamento de projetos e *bugs*) e o SVN (sistema de versionamento de código e artefatos). Esse processo foi utilizado por diversos projetos do LEDES, em especial, o projeto do SIGFAP, que atualmente está sendo utilizado em quinze[3] estados do país. O ProFap é composto por seis etapas, conforme apresentadas na Figura 1.

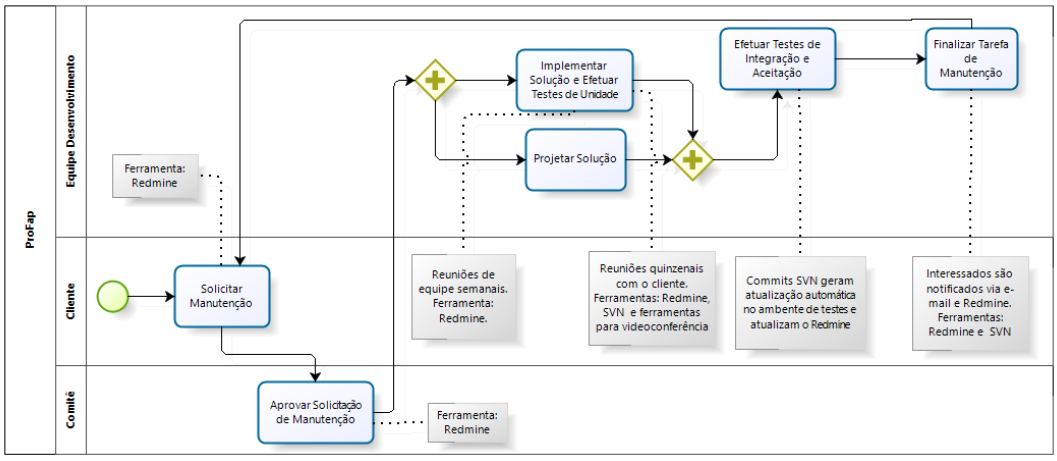

Figura 1. Processo ProFap: Processo Colaborativo de Manutenção (CAGNIN *et al*., 2013)

Na etapa "Solicitar Manutenção", é feita a solicitação pelo cliente via Redmine de novas funcionalidades do sistema ou o registro de *bugs*. Se a solicitação for relativa a manutenção corretiva, então será atribuída a um desenvolvedor. Se referente ao desenvolvimento de módulo ou funcionalidade inédita, então será avaliada por comitês na etapa "Aprovar Solicitação de Manutenção". Nessa etapa, uma solicitação pode ser aprovada ou cancelada.

Na etapa "Projetar Solução", após aprovação de uma solicitação, esta é atribuída a um desenvolvedor. Nela, desenvolvedores com tarefas atribuídas relacionadas entre si discutem a melhor forma de projetar a solução para atender tais tarefas, com auxílio dos líderes de equipe que possuem conhecimento detalhado do domínio do sistema. Em paralelo à etapa "Projetar Solução", ocorre a etapa "Implementar Solução e Efetuar Testes de Unidade". Testes funcionais e unitários devem ser feitos pelo desenvolvedor,

[3] AC, AL, AM, AP, DF, ES, MS, MT, PA, PI, PR, RO, RS, SE e TO.

em seu ambiente local, durante a implementação da solicitação. Durante essas etapas, pode ocorrer, via Redmine, comunicação entre cliente e desenvolvedor.

Na etapa "Efetuar Testes de Integração e de Aceitação" é feita atualização do ambiente de testes, via SVN integrado ao Redmine, com o resultado do trabalho. Após isso, são realizados testes de integração no ambiente de testes. O teste de aceitação é sempre feito pelo cliente. Por fim, na etapa "Finalizar Tarefa de Manutenção", ocorre o encerramento da solicitação de manutenção via Redmine e uma nova versão do sistema é atualizada no ambiente de produção utilizando o SVN.

Em maio de 2015, com a implantação do papel gerente de projeto no projeto SIGFAP, foram detectadas diversas deficiências no processo ProFap: quanto a processos: ProFap não era cumprido, as informações disponíveis no Redmine não eram confiáveis pois os membros da equipe se esqueciam de informar dados importantes como total de horas gasto em cada tarefa, funcionalidades incompletas eram enviadas para produção por engano; quanto aos papéis: líderes de equipe e equipe de manutenção sem responsabilidades bem definidas; quanto a tarefas: os próprios clientes criavam as tarefas no Redmine, isso culminava em muitas tarefas sem necessidade, tarefas duplicadas, tarefas esquecidas e tarefas sem acompanhamento; quanto ao desenvolvimento: ausência de análise, estimativa e planejamento das tarefas, ausência de testes; quanto a comunicação: clientes com acesso direto à equipe de manutenção, uso de diversos canais de comunicação interna e externa (Hangout, Whatsapp, Messenger, Skype, etc), levando a perda do histórico de comunicação.

3. ProLedes: Processo de Desenvolvimento Colaborativo de Software do Ledes

A partir das deficiências detectadas no processo ProFap e apresentadas na seção anterior, algumas melhorias foram propostas e implantadas em junho/2015: definição dos processos Gerência de Projeto, Suporte, Análise, Desenvolvimento, Teste; definição dos papéis Gerente de Projeto, Suporte, Analista de Sistemas, Desenvolvedor, Testador e definição de suas respectivas responsabilidades; definição dos *tickets* de solicitação (que representa a necessidade do cliente) e de tarefa; definição de regras (suporte gera solicitação e, a partir da solicitação, analista de sistemas pode ou não gerar tarefa, que será analisada, projetada, implementada, testada e implantada em produção), personalização do Redmine para atender os processos, papéis e seus privilégios, e *tickets* definidos; implantação da ferramenta Slack[4] como ferramenta de comunicação, cliente com acesso limitado ao suporte, analista de sistemas e gerente de projeto, separação da comunicação interna e externa em grupos distintos no Slack e toda a comunicação interna e externa passou a ser preferencialmente pública entre os membros do projeto.

Após o primeiro mês de implantação das melhorias supracitadas (julho/2015) foram encontradas muitas resistências internas (por exemplo, alguns membros não queriam assumir os novos papéis e atribuições) e externas (por exemplo, discordância dos clientes por terem perdido acesso aos desenvolvedores). Nesse momento, como o

[4] https://slack.com/

SIGFAP não possuía documentação, houve necessidade de definir o processo Documentação para iniciar a condução da Engenharia Reversa (CHIKOFSKY e CROSS, 1990). Como consequência, foi definido o papel Analista de Requisitos e suas responsabilidades.

Em setembro/2015, ainda havia resistência interna, e alguns membros da equipe foram substituídos e outros foram promovidos (por exemplo, de desenvolvedor para analista de sistemas). Em outubro/2015 ocorreram mudanças na composição da equipe (conforme esperado em ambiente acadêmico), principalmente com a saída de um desenvolvedor com muita experiência, o que culminou em um nível mínimo de produtividade da equipe.

Em novembro/2015, houve seleção e recrutamento de três novos desenvolvedores. Para alcançar a produtividade desejada em um curto espaço de tempo, foi dado treinamento intensivo até dezembro/2015 a esses novos desenvolvedores tanto no processo quanto nas tecnologias utilizadas no projeto SIGFAP com apoio principalmente da prática programação em pares (BECK e ANDRES, 2004) para alcançar, em curto prazo, nivelamento do conhecimento entre os membros da equipe. Nesse momento, houve intenção de integrar o processo Documentação aos demais processos (Suporte, Análise, Desenvolvimento, Teste), pois os documentos estavam sendo elaborados pelo Analista de Requisitos mas ainda não estavam sendo utilizados pela equipe.

Por se tratar de um processo de manutenção em um laboratório de pesquisa que possui alta rotatividade de alunos e desenvolvedores, a documentação tornou-se fundamental para garantir a continuidade do projeto e a realização de alterações no código de forma segura. Durante a manutenção, a documentação ocorre após a tarefa de desenvolvimento para garantir a consistência da documentação com a alteração realizada no código. Antes da adoção do processo Documentação, o gerente do projeto ficava responsável por responder todos os questionamentos da equipe, causando sobrecarga. Além disso, não era possível prever quais efeitos uma mudança poderia provocar no projeto.

Em maio/2016 observou-se que a equipe estava estabilizada. Nesse momento, houve promoção de um desenvolvedor para analista de requisitos, a substituição da ferramenta SVN pela ferramenta Git[5], pois acredita-se que essa ferramenta oferece mais facilidades para apoiar a refatoração. Houve também a disponibilização da documentação do SIGFAP[6] a todos os membros da equipe.

Para dirimir o impacto da rotatividade da equipe, houve treinamento e motivação constante para que todos pudessem contribuir efetivamente com o projeto. Em especial, a motivação foi baseada em elogiar publicamente os membros da equipe, advertir apenas em particular, incentivar desafios e maximizar os acertos (WRIGHT, 2012).

Após a implantação das melhorias, observou-se estabilidade nos processos e papéis definidos, bem como na comunicação e tecnologias adotadas. Além disso, ressalta-se

[5] https://git-scm.com/
[6] http://sigfap.ledes.net/docs/

que não houve aumento de recursos humanos durante a implantação das melhorias. No entanto, observou-se que a reorganização em papéis aumentou a eficiência da equipe.

Na Figura 2 é apresentado o ProLedes, resultante da evolução do ProFap. O cliente inicia todo o processo, solicitando suporte via ferramenta Slack. A equipe de suporte atende o cliente e registra no Redmine o atendimento prestado. Caso o atendimento não resolva o problema, uma solicitação é registrada no Redmine. Essa solicitação é analisada pela equipe de analistas de sistemas. Caso a solicitação seja rejeitada por não tratar de uma correção, refatoração ou implementação de uma nova funcionalidade dentro do escopo do sistema de software, encerra-se o fluxo do processo. Por outro lado, cria-se uma tarefa de acordo com o seu objetivo. No início da semana, toda equipe reúne-se para a reunião de planejamento semanal a fim de planejar as tarefas que serão desenvolvidas. A equipe de desenvolvimento se responsabiliza por desenvolver e entregar uma solução para as tarefas alocadas na semana corrente. A entrega da solução é feita para a equipe de teste que realiza testes funcionais com base na descrição das tarefas. A solução é implantada em ambiente de produção caso atenda a solicitação. Por outro lado, a solução retorna para a equipe de desenvolvimento realizar os ajustes necessários de acordo com o *feedback* da equipe de teste. Finalmente, o analista de requisitos gerencia a documentação elaborando ou atualizando casos de uso relacionados à tarefa. Todas as atividades do processo são monitoradas e controladas pelo gerente de projeto.

Atualmente somente a equipe do Ledes utiliza o ProLedes, sendo o principal projeto o do sistema SIGFAP que é utilizado por 15 FAPs do Brasil e outras Fundações: Araucária, CONFAP, FAPAC, FAPDF, FAPEAL, FAPEAM, FAPEAP, FAPEMAT, FAPEPI, FAPERGS, FAPERO, FAPES, FAPESPA, FAPITEC, FAPT, FUNDECT, SECTEI, SETI. Em cada FAP existe, no mínimo, um representante que é responsável pela comunicação entre a FAP e a equipe de desenvolvimento do Ledes.

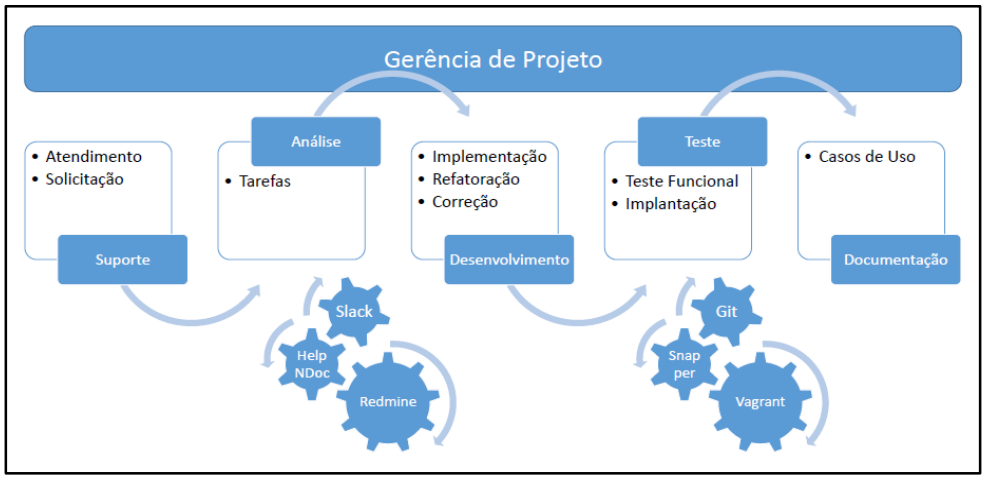

Figura 2. Processo ProLedes[7]

[7] Documentação completa disponível em: http://sigfap.ledes.net/docs/Processos_ProLEDES_v1_3.pdf

3.1 Artefatos

Durante o uso do ProLedes são elaborados os seguintes artefatos: i) ***tickets* de atendimento, solicitação e tarefa**. Os *tickets* de tarefas podem ser do tipo correção, implementação, refatoração, documentação, treinamento e reunião. Os *tickets* contêm informações relevantes que apoiam o desenvolvimento e a manutenção de software como: descrição detalhada, responsável (membro da equipe), tempo estimado e tempo gasto, complexidade (alta, média ou baixa), prioridade (baixa, normal, baixa e urgente) e *link* para documentação de apoio; ii) **documentação** das funcionalidades no formato de descrição expandida de **casos de uso** e iii) **código fonte**. Os *tickets* passam por vários *status* durante o uso do ProLedes. Isso garante o acompanhamento, pelo cliente, do atendimento de uma solicitação ou de uma tarefa. Os possíveis *status* de uma solicitação são: nova, em análise, *feedback* (quando é necessário elicitar requisitos com o cliente), análise pronta (que pode gerar ou não *tickets* de tarefa), fechada (depois que as tarefas relacionadas forem fechadas) e rejeitada. Os possíveis *status* de uma tarefa são: em análise, *feedback*, análise pronta, em execução (quando a tarefa foi atribuída pelo gerente de projeto), pronta, em teste, em documentação e fechada.

3.2. Papéis

Os papéis do ProLedes são: i) **diretoria:** gestores dos acordos e termos de cooperação da Rede SIGFAP. Mantém plano estratégico, toma decisão colegiada sobre questões diversas, aprova ação planejada pelo gerente de projeto e coordena acordos/termos de cooperação; ii) **cliente:** colaborador da FAP responsável por representá-la na Rede SIGFAP. Solicita ao gerente de projeto agendamento de reunião ou treinamento e negocia a prioridade das solicitações e tarefas; iii) **usuário:** pesquisador ou colaborador da FAP. Solicita atendimento pelo suporte técnico e solicita mudança ou correção do sistema; iv) **suporte:** presta suporte técnico via Slack. Auxilia o cliente nas atividades de uso do sistema, registra como atendimento o suporte técnico prestado ao cliente e registra como solicitação as necessidades do cliente (solicitações de mudança ou correção); v) **gerente de projeto:** decide, monitora e controla o processo de desenvolvimento. Rejeita solicitação e tarefas, conduz o planejamento, atribui tarefa, decide escopo e prioridade de tarefas em conjunto com o cliente e a equipe de desenvolvimento, e motiva a equipe para utilizar efetivamente o ProLedes e para trabalhar com foco na qualidade; vi) **analista de sistemas:** realiza treinamentos, analisa solicitações, projeta tarefas, sugere escopo e estimativa das tarefas antes do planejamento; vii) **analista de requisitos:** elicita requisitos, elabora ou atualiza descrições expandidas de casos de uso usando o HelpNDoc[8] ou descrições de tarefas, valida tarefas e vincula documentação relevante; viii) **desenvolvedor:** implementa uma solução para tarefas de correção, implementação ou refatoração; ix) **testador:** testa a solução com base na descrição da tarefa. Se a solução fornecida atender a solicitação do cliente, então aceita e implanta, senão rejeita e notifica o defeito. Salienta-se que todos os atores do ProLedes geram evidências do esforço de trabalho desempenhado em sua rotina, lançando horas de trabalho nos *tickets* do Redmine.

[8] http://www.helpndoc.com/

3.3. Atividades

São apresentados a seguir os principais processos, com suas atividades correspondentes, que compõem o processo ProLedes.

3.3.1. Gerência de Projetos

Este processo tem por objetivo contribuir com o planejamento, tomada de decisões, monitoramento, controle e melhoria do projeto como um todo. Suas principais atividades são: rejeitar solicitação e tarefa (a tarefa de aceite da solicitação é realizada pelo analista de sistemas e, caso ele opte pela rejeição, o gerente analisa a situação e confirma a rejeição), atribuir e priorizar tarefas, fazer o planejamento semanal, agendar e realizar reunião com os membros e agendar treinamento para os usuários.

A rejeição da solicitação é negociada com o cliente, podendo ser uma decisão unilateral da gerência, de acordo com os recursos disponíveis. Na reunião de planejamento semanal os recursos necessários são alocados. Durante a reunião, o gerente de projeto faz a retrospectiva da reunião anterior e atualiza o último planejamento. O gerente de projeto atribui tarefas aos desenvolvedores, incluindo-as em um planejamento. As estimativas referentes a cada tarefa são realizadas utilizando *planning poker*. Ao final, as ferramentas de gerenciamento são atualizadas e o tempo gasto para a reunião é registrado.

Na priorização de tarefas são considerados fatores como: (a) tarefas com prioridade normal são atendidas preferencialmente por ordem de registro; (b) o cliente pode solicitar a alteração da prioridade da análise de uma solicitação ou execução de uma tarefa para urgente e (c) o custo de uma urgência é no mínimo dobrado e impacta na produtividade e no planejamento.

As reuniões (presenciais ou virtuais) ocorrem de acordo com agendamento prévio e têm como principal objetivo a discussão dos itens da pauta. As principais reuniões são: i) reunião de planejamento semanal, que ocorre uma vez por semana, e ii) reunião técnica, que ocorre com uma frequência média de uma a cada duas semanas, mas é agendada previamente entre os desenvolvedores quando um assunto técnico precisa ser discutido. O treinamento ocorre de acordo com as solicitações dos clientes e com a disponibilidade de recursos da equipe de analistas e desenvolvedores. Todo treinamento é previamente agendado. O tempo gasto é lançado conforme a atividade. As principais ferramentas de apoio à Gerência de Projetos são: Slack, Redmine, HelpNDoc e Skype.

3.3.2. Suporte

Este processo visa prestar suporte técnico e gerar registros do atendimento, das solicitações e dos eventos anômalos ou errôneos do sistema. Sua principal atividade é prestar suporte técnico a clientes e usuários. Neste processo o cliente ou usuário se comunica com o suporte utilizando a ferramenta Slack por meio de um canal específico. Se o problema foi resolvido com o auxílio do suporte técnico, um atendimento é registrado. Senão, é aberta uma solicitação e o gerente de projetos define qual a sua prioridade. Durante a análise da solicitação pelo analista de sistemas, tarefas são criadas para atendê-la. Finalmente, depois da execução das tarefas, ela é fechada. O tempo

gasto na atividade pelo suporte é lançado. As principais ferramentas de apoio ao suporte são: Slack, Redmine, HelpNDoc.

3.3.3. Análise

Este processo visa analisar e projetar as tarefas a serem executadas. Suas principais atividades são: analisar solicitações e gerar as tarefas relacionadas, analisar e projetar as tarefas, preparar e realizar treinamentos. Inicialmente, a solicitação é analisada e, caso seja pertinente, é aceita. O analista de sistemas projeta uma solução, registrando os detalhes da modelagem em uma ou mais tarefas. Então, o analista de requisitos verifica cada tarefa e anexa a documentação existente. Ao preparar treinamento, gera o material didático necessário. Conforme o agendamento, o analista de sistemas realiza o treinamento por meio de reunião (presencial ou virtual) entre os participantes. O tempo gasto nas atividades de análise é lançado. As ferramentas utilizadas são: Slack, Redmine, HelpNDoc, Snapper e Vagrant.

3.3.4. Desenvolvimento

Este processo visa executar as tarefas e gerar as soluções do sistema, sendo bastante dependente do uso de ferramentas de apoio. O desenvolvedor gera uma *branch* para a tarefa, usando a ferramenta Git, associada ao *ticket* da tarefa do Redmine. Essa associação garante a rastreabilidade bidirecional devido ao fato do *ticket* no Redmine ter *link* para a documentação produzida no HelpNDoc. O desenvolvedor também gera um banco de dados descaracterizado usando o Snapper, simula um ambiente para desenvolvimento usando o Vagrant. As mudanças são mantidas na *branch* criada para a tarefa até sua finalização. Por fim, o desenvolvedor detalha a solução e lança o tempo gasto nas atividades correspondentes. Alternativamente, o desenvolvedor pode rejeitar uma tarefa devido a descrição insuficiente, o que faz com que a tarefa retorne para o processo Análise. As ferramentas utilizadas são: Slack, Redmine, HelpNDoc, Snapper, Git e Vagrant.

3.3.5. Teste

Este processo visa testar, aceitar e implantar as soluções do sistema. O testador obtém a *branch* correspondente à tarefa que está sendo testada usando a ferramenta Git, simula o sistema usando Vagrant, valida e aceita a solução e mescla a *branch* com a *master* (*baseline*). Em seguida os servidores automaticamente implantam a solução por meio de um *script* de *deploy*, e o testador encaminha a tarefa para documentação. O tempo gasto nas atividades deste processo é lançado. As ferramentas utilizadas são: Slack, Redmine, HelpNDoc, SubGit, SVN, Git e Vagrant.

3.3.6. Documentação

Este processo visa gerar, disponibilizar e manter os artefatos de documentação do sistema. De acordo com as necessidades dos projetos, são utilizados *templates* para a descrição das tarefas. Outros artefatos de documentação necessários, tais como casos de uso estendidos e regras de negócios, são produzidos usando a ferramenta HelpNDoc. Quando ocorre alguma modificação no sistema, os artefatos de documentação também são atualizados e novas versões são geradas. O tempo gasto nas atividades de

documentação é lançado. *Logs* de mudanças das tarefas são mantidos. São utilizadas as ferramentas: HelpNDoc, Slack, Redmine, HelpNDoc, Vagrant, Snapper e Git.

3.4 Personalização das Ferramentas de apoio

3.4.1. Redmine

O Redmine é uma aplicação Web personalizável e de código aberto para gerência de projetos. Oferece suporte a múltiplos projetos, controle de acesso flexível baseado em papéis, integração com ferramentas de gerência de configuração e de comunicação. O Redmine foi reconfigurado para se adequar ao fluxo de trabalho, papéis e *tickets* definidos no ProLedes, com o intuito principal de controlar e acompanhar a execução dos atendimentos, solicitações e tarefas, gerando assim evidências do uso do ProLedes. Para facilitar a gestão das solicitações e tarefas de todas as FAPs, foram criados um projeto para representar toda a Rede SIGFAP e um subprojeto para cada FAP da rede. As tarefas são gerenciadas no projeto principal e as solicitações e atendimentos em cada subprojeto correspondente. Foi criado um grupo de usuários para cada papel definido, menos o papel diretoria, que tem as mesmas permissões que o gerente de projeto e pertencem ao mesmo grupo. Para cada grupo foram concedidas as permissões necessárias para a execução das responsabilidades do papel correspondente. Os *tickets* foram personalizados em três tipos: atendimento, solicitação e tarefa. Para os *tickets* foram criados todos os campos necessários para gerar as evidências de trabalho desejadas, inclusive os *status* definidos para cada tipo de *ticket*. Os *status* dos *tickets* possibilitaram a definição de um fluxo de trabalho que vincula os papéis às suas respectivas responsabilidades.

3.4.2. Slack

O Slack é uma aplicação Web para comunicação de times. Oferece funcionalidades de comunicação em tempo real, arquivamento e busca, tornando possível centralizar toda a comunicação em um só lugar. Além disso, notifica via *email* os membros que estiverem *offline*. O ProLedes passou a adotá-lo com objetivo principal de facilitar o gerenciamento da comunicação interna e externa em um único lugar. Dois times foram criados no Slack: LEDES, para a comunicação interna, e SIGFAP, para a comunicação externa.

3.4.3 Git

O Git é um sistema de controle de versões distribuído que facilita o gerenciamento e o controle da evolução do código fonte de um software desenvolvido por meio de múltiplas *branches*. O ProLedes passou a adotá-lo com objetivo principal de reduzir incidentes pós-atualização, mas também para alcançar a rastreabilidade bidirecional entre requisito de sistema e código fonte.

O ProLedes lida com uma dificuldade comumente enfrentada em processos colaborativos de manutenção de software, que é conseguir isolar as modificações feitas para atender cada mudança e conseguir atualizá-las corretamente nos ambientes de produção, mesmo quando há dependências de partes sendo trabalhadas em outras tarefas. As instâncias de produção dos softwares mantidos usando o ProLedes têm seu código fonte idêntico a uma *branch* do repositório Git do projeto, denominada *branch*

master. Com o uso das *branches* associadas a cada *ticket* de tarefa, os testadores conseguem avaliar o impacto de cada mudança de maneira isolada na *branch master*. Essa prática eliminou problemas que ocorriam quando partes de código sendo trabalhadas eram enviadas para as instâncias de produção por engano, reduzindo consideravelmente os incidentes pós-atualização.

No ProLedes, o Git é utilizado de maneira integrada ao Redmine permitindo a rastreabilidade entre o código e a documentação da tarefa a ser realizada. Essa prática, juntamente com a correta adoção do HelpNDoc para documentação de requisitos, viabiliza a rastreabilidade bidirecional entre os requisitos de sistema e o código fonte. Durante as tarefas relacionadas a *tickets* de implementação, correção e refatoração, os desenvolvedores realizam *commits* do código em suas *branches* específicas de desenvolvimento. Ao realizar um *commit*, o desenvolvedor adiciona uma breve mensagem explicando as mudanças realizadas naquele *commit* e, nessa mensagem, informa o identificador do *ticket* de tarefa que está registrado no Redmine. Isso garante rastreabilidade entre os *tickets* e o código fonte.

3.4.4. HelpNDoc

O HelpNDoc foi adotado como ferramenta para centralizar a especificação do sistema do projeto. Essa ferramenta permite a edição de documentos de texto técnicos e viabiliza a exportação dos documentos no formato HTML para disponibilização na Web. As páginas e seções dos textos escritos nessa ferramenta podem ser relacionadas entre si e indexadas para facilitar buscas. Esta ferramenta completa a rastreabilidade bidirecional entre requisitos e código fonte. No portal de documentação de um projeto que utiliza o ProLedes, são documentados os requisitos do software, casos de uso e regras de negócios. Cada página da documentação é publicada na Web e associada a *tickets* de tarefas no Redmine, relacionados à manutenção e evolução do sistema. Essa associação permite a visualização clara do impacto no código fonte das mudanças de regras de negócio e novas necessidades de clientes, bem como, quais requisitos são influenciados por refatorações realizadas pelos desenvolvedores.

3.4.5. Vagrant

O Vagrant[9] permite a automatização na configuração de ambientes de desenvolvimento baseados em máquinas virtuais. As atividades de desenvolvimento e teste do ProLedes exploram essa ferramenta para eliminar a complexidade e o esforço de configuração de ambiente de desenvolvimento. Quando um novo desenvolvedor entra na equipe, ele pode imediatamente começar a contribuir com tarefas e aprender já na prática, pois, todos os softwares necessários para suas atividades já são fornecidos e configurados em um ambiente padrão desenvolvido pela equipe de infraestrutura do projeto. Além de economizar tempo na configuração do ambiente de desenvolvimento local, Vagrant garante ao desenvolvedor um ambiente para desenvolvimento muito próximo dos ambientes de produção, ou seja, um ambiente com as mesmas versões de software e configurações de rede e de programas bem similares. Essa similaridade garante uma economia de tempo em buscas por causas de problemas que ocorrem nos ambientes de produção em situações específicas.

[9] https://www.vagrantup.com/

3.4.6. Snapper

O Snapper é uma ferramenta desenvolvida em *Bash Linux* e mantida pelos autores do ProLedes que complementa o ambiente virtual de desenvolvimento do Vagrant, automatizando mudanças em arquivos de configuração e nas bases de dados para simular diferentes instâncias de produção do sistema. Alguns problemas reportados por clientes que ocorrem nos ambientes de produção são mais difíceis de serem reproduzidos nos ambientes de desenvolvimento por dependerem de dados específicos do sistema de produção. Ao executar um *script* Snapper, o desenvolvedor informa um nome, que representa a instância de produção a ser simulada. O *script* realiza o *download* da base de dados dessa instância de produção configurando-a no ambiente de desenvolvimento local. Com o intuito de manter a privacidade dos dados de usuários do sistema, o *script* Snapper roda uma rotina de descaracterização, que altera os dados pessoais dos usuários, como e-mail, CPF, endereço. Com acesso a um ambiente simulando um estado muito próximo do ambiente real onde o problema foi reportado, e com as ferramentas de desenvolvimento a sua disposição, o desenvolvedor consegue rapidamente identificar a causa de problemas e projetar uma solução.

4. Estudo de caso: evidências de melhorias com o ProLedes

O **objetivo** do estudo de caso é evidenciar as melhorias obtidas com o processo ProLedes. Esse objetivo foi detalhado de acordo com Wohlin *et al*. (2012): **analisar** os registros do uso do ProLedes com base nas ferramentas utilizadas; e também **analisar** as opiniões dos clientes **com o propósito de** levantar evidências das melhorias **com relação às** iniciativas de melhoria em processo de software do laboratório de pesquisa LEDES **sob o ponto de vista de** engenheiros de software e clientes **no contexto do** projeto SIGFAP. Na Tabela 1 são apresentadas as deficiências do ProFap e as iniciativas de melhorias realizadas para dirimi-las.

Tabela 1. Evidências de melhorias com base em registros das ferramentas

Deficiências	Descrição das evidências de melhoria
Processo de desenvolvimento e manutenção não utilizado pela equipe	Por meio do uso do Redmine pelos atores do processo, garantiu-se o uso do processo. Na ferramenta é possível observar as horas trabalhadas e as descrições detalhadas do esforço realizado, tanto nas solicitações quanto nas tarefas.
Informações não confiáveis sobre o projeto	Com o uso efetivo do Redmine e também com o uso do HelpNDoc é possível produzir e manter informações confiáveis sobre o projeto. O HelpNDoc evidencia a documentação atualizada de requisitos do sistema e a integração do processo de documentação no fluxo de tratamento de solicitações e tarefas, por meio da rastreabilidade com os demais artefatos do processo.
Membros da equipe sem responsabilidades bem definidas	O Gerente de Projetos monitora e controla as atividades do ProLedes, garantindo que os demais atores do ProLedes desempenhem suas atividades cumprindo suas responsabilidades.
Tarefas sem necessidade, tarefas duplicadas, tarefas esquecidas, tarefas sem acompanhamento, tarefas "sem fim"	O ator Suporte foi introduzido no processo para mitigar esse problema. As solicitações do cliente são feitas ao Suporte via Slack e são registradas adequadamente no Redmine.

Ausência de análise, estimativa e planejamento das tarefas	O ator Analista de Sistemas foi introduzido no processo para ajudar a mitigar esse problema. Adicionalmente, na reunião semanal de planejamento as tarefas são estimadas e planejadas pela equipe.
Ausência de testes	Todas as tarefas com *status* "pronto" passam pelo processo Teste. Os requisitos descritos na tarefa são testados antes da atualização do sistema em produção.
Pouca produtividade dos desenvolvedores	Com a introdução dos atores Suporte, Analista de Sistemas e Gerente de Projetos, os Clientes não têm mais acesso ao Desenvolvedor, o que permite o seu foco na implementação das soluções. Por meio dos registros de horas trabalhadas no Redmine foi possível observar um aumento de produtividade por parte dos desenvolvedores.
Ausência de histórico das comunicações internas e externas	A ferramenta Slack forneceu registro do histórico das comunicações rotineiras do uso do processo e, em conjunto com o registro de *tickets* do Redmine, o histórico das comunicações foi viabilizado.
Falta de documentação do SIGFAP	O processo Documentação foi introduzido no processo, bem como o ator Analista de Requisitos, que é o responsável por manter a documentação atualizada e integrada com os demais artefatos.
Rotatividade da equipe	Todos contribuem efetivamente com o projeto por meio do registro das evidências nas ferramentas de apoio utilizadas pelo ProLedes. Isso é alcançado devido a treinamentos e motivação constantes.
Partes de código sendo trabalhadas eram enviadas para as instâncias de produção por engano	A implantação do Git viabilizou a organização das atividades em *branches* de desenvolvimento que permitiu o isolamento das modificações, eliminando o engano no envio das mudanças para produção.

Com o registro das solicitações e tarefas no Readmine, foi possível levantar, de maneira quantitativa, os benefícios obtidos com a implantação das iniciativas de melhoria realizadas no período de agosto/2015 a julho/2016, conforme apresentado na Tabela 2 e no gráfico da Figura 3. A partir dos dados apresentados observa-se aumento significativo da produtividade da equipe, mesmo com o recesso de 15 dias no mês de dezembro de 2015.

Tabela 2. Evidências sobre o aumento da produtividade da equipe

	Ago-Out (Em treinamento)	**Nov-Mar (Alta rotatividade)**	**Abr-Jul (Alto desempenho)**
Tempo médio (em dias) entre solicitação e término da análise	4,75	4,04 (com analista experiente)	4,25 (com analista novo)
Qtde de solicitações analisadas em até dois dias corridos	70	111	128

Para levantar a opinião dos clientes quanto as melhorias com a implantação do ProLedes, foi elaborado um questionário no GoogleForms com 12 questões objetivas e com respostas baseadas na escala de Likert. O *link* para o questionário foi divulgado no Slack para todos os clientes da Rede SIGFAP. Dos 18 representantes da rede, 15 responderam o questionário perfazendo 83% dos clientes. A partir da análise das respostas, 93,3% estão satisfeitos com o ProLedes, 80% perceberam que os processos e os papéis estão bem definidos, 73,3% perceberam transparência na execução do processo, 100% confiam nas informações de acompanhamento de solicitações e de tarefas e 80% percebem que suas solicitações são priorizadas adequadamente. Além disso, a utilização do Slack como ferramenta de comunicação foi observada como um

dos principais fatores de sucesso na resolução dos problemas identificados. Apesar dos bons resultados levantados, melhorias precisam ser implantadas para aprimorar a percepção de produtividade na execução das tarefas e, também, evidenciar os testes realizados e a implantação nos servidores de produção.

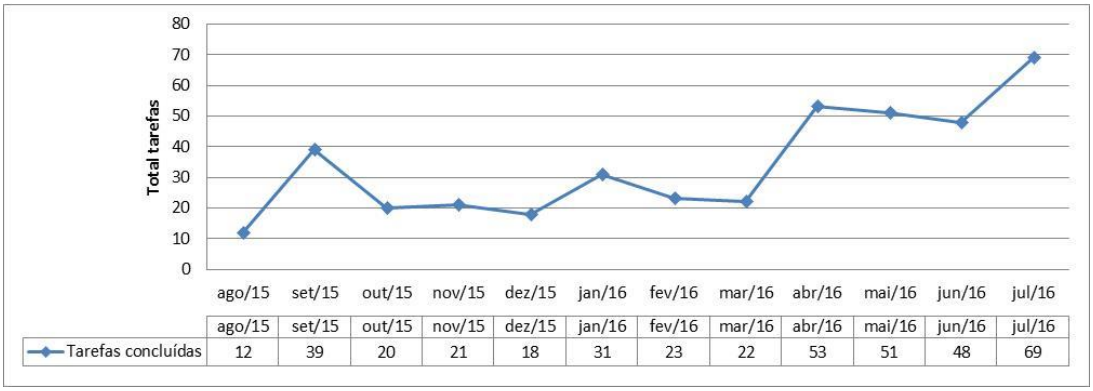

Figura 3. Tarefas concluídas mensalmente pela equipe

5. Conclusão

Este artigo apresentou os problemas existentes no processo ProFap e as iniciativas de melhorias adotadas para contornar esses problemas obtendo-se o processo ProLedes. Essas iniciativas de melhorias foram propostas conforme as necessidades do LEDES, em especial do projeto SIGFAP, e foram baseadas em práticas, métodos e técnicas de Engenharia de Software (SOMMERVILLE, 2016; BECK e ANDRES, 2004; SCHWABER e BEEDLE, 2003), bem como em práticas de motivação pessoal (WRIGHT, 2012). Evidências das melhorias implantadas com o uso do ProLedes no projeto SIGFAP foram coletadas a partir dos registros dos dados das ferramentas de apoio, bem como com base em opiniões dos clientes. A partir das experiências obtidas com a implantação das melhorias, algumas lições aprendidas podem ser destacadas: utilizar programação em pares em equipes com diferentes níveis de conhecimento para nivelar o conhecimento em um curto espaço de tempo; reuniões semanais de planejamento em equipe são úteis para estimar as tarefas e manter o comprometimento da equipe; ferramentas computacionais são primordiais para gerenciar e controlar o uso efetivo de processos de software; evitar comunicação entre clientes e desenvolvedores para não prejudicar a produtividade; comunicação pública entre os membros do projeto com apoio de ferramenta propicia a transparência e a agilidade das comunicações entre os papéis e favorece a produtividade da equipe; oferecer treinamento e manter a equipe motivada reduz o impacto que ocorre com a rotatividade da equipe.

Levando em consideração o grande avanço obtido com a implantação do ProLedes, a equipe do LEDES está planejando a reengenharia do SIGFAP como trabalho futuro para facilitar o atendimento das solicitações dos diversos clientes, uma vez que o SIGFAP é um sistema legado desenvolvido desde 2001. Além disso, serão planejadas iniciativas de melhorias no ProLedes para mitigar as deficiências existentes indicadas pelos clientes no questionário de avaliação do processo.

Referências

BECK, K.; ANDRES, C. (2004) Extreme Programming explained: embrace change. 2nd edition, Addison-Wesley.

CAGNIN, M. I., TURINE, M., SILVA, M., LANDRE, G., OLIVEIRA, L., LIMA, V., SANTOS, M., PAIVA, D., CARROMEU, C. (2013) ProFap: Processo Colaborativo de Manutenção de Software. In: X WMSWM, Salvador, BA.

CARROMEU, C., PAIVA, D. M. B., CAGNIN, M. I., RUBINSZTEJN, H. K. S., TURINE, M. A. S., BREITMAN, K. (2010) Component-Based Architecture for e-Gov Web Systems Development. In: 17th ECBS, Oxford, UK.

CHIKOFSKY, J. E., CROSS, J. H. (1990) Reverse engineering and design recovery: A taxonomy. IEEE Software, v. 7, n. 1, p. 13-17.

FUGGETA, A. (2000) Software Process: A RoadMap. In: 22nd ICSE, Ireland, p. 25-34.

ISO/IEC (2003) ISO/IEC 15504: Information Technology Process Assessment. International Standard.

MENDES, F. F.; FERNANDES, P. G.; MOTA, C. C.; NUNES, R. S., MARTINS, M. D. S. (2011) Institucionalização de Processos no Setor de Produção de Software de uma Universidade Federal. In: X SBQS, Curitiba, PR, p. 369-376.

MENDES, F. F.; MESQUITA, H. F.; ALMEIDA, J. N. (2012) A Influência do Processo, Pessoas e Tecnologia na Qualidade de Software: um Estudo de Caso. In: XI SBQS, Fortaleza, CE, p. 83-97.

PFLEEGER, S. L. (2004) Engenharia de Software: Teoria e Prática. São Paulo: Prentice Hall.

PRESSMAN, R. S. (2016) Engenharia de Software. São Paulo: Bookman.

SANTOS, I.S.; FRANCO, W.; SABÓIA, B.; ANDRADE, R. M. C. (2015) Definição e Aplicação de um Processo de Testes Ágeis: um Relato de Experiência. In: XIV SBQS, Manaus, MA, p. 228-235.

SCHWABER, K.; BEEDLE, M. (2003) Agile Software Development with SCRUM, Prentice-Hall.

SEI (2010) CMMI® for Development (CMMI-DEV) - Improving processs for developing better products and services, V 1.3, CMU/SEI-2010-TR-033, Software Engineering Institute.

TURINE, M. A. S.; CARROMEU, C.; SILVA, M. A. I.; CAGNIN, M. I. (2011) Gestão pública flexível e ágil por meio do SIGFap. Revista eletrônica de jornalismo científico ComCiência.

VIANA, D.; SOUZA, C.S.; CABRAL, R.; DIB, M.; VIEIRA, A.; FERREIRA, R.; CONTE, T. (2014) Usando Análise de Redes Sociais para Investigar Disseminação do Conhecimento em Melhorias de Processos de Software. In: XIII SBQS, Blumenau, SC, p. 179-193.

WRIGHT, J. (2012) Teacher praise: An efficient tool to motivate students. White Paper, Disponível em: www.interventioncentral.org. Acesso em junho/2016.

WOHLIN, C., RUNESON, P., HST, M., OHLSSON, M., REGNELLl, B., WESSLN, A. (2012) Experimentation in Software Engineering. Springer, Heidelberg.