An Agile Approach Applied to Intense Maintenance Projects
Uma Abordagem Ágil Aplicada a Projetos de Manutenção Intensa

Gibeon Soares de Aquino Junior
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
gibeon@dimap.ufrn.br

André Medeiros Dantas
Superintendência de Informática
Universidade Federal do Rio Grande do Norte
andredantas@info.ufrn.br

RESUMO

Esse artigo propõe uma abordagem para gerenciar projetos de software susceptíveis a constantes e intensas solicitações de mudanças que requerem um rápido tempo de resposta. Muito influenciado pelo pensamento ágil de desenvolvimento de software, em especial SCRUM, Kanban e Lean, essa solução captura a filosofia e principais técnicas desses "métodos" ágeis. Além disso, inclui o uso de medidas específicas para gerenciar a saúde do projeto e prover informações precisas para a alta gestão da organização. O foco dessa proposta se encontra na otimização e melhoria do processo de manutenção de sistemas de informação de grande porte que precisam atender diariamente solicitações de mudanças de milhares de usuários. A aplicação dessa abordagem em um estudo de caso real permitiu adquirir resultados significativos que indicam o potencial da proposta.

PALAVRAS CHAVE

Manutenção de software; Evolução de Software; Sistemas de Informação; Métodos ágeis.

ABSTRACT

This paper proposes an approach to managing software projects that are susceptible to frequent and intense change requests demanding quick response time. Highly influenced by agile thinking, in particular, SCRUM, Kanban and Lean Software development, this solution directly captures the philosophy and the main techniques of these popular agile "methodologies". Moreover, it includes the use of specific measures to manage the project's health and give accurate information to the top level management. This proposal's focus is on improving and optimizing the maintenance process of large-scale information systems which daily handle the change requests of thousands of users. The application of this approach in a real study case allowed to obtain significant results that indicate the potential of the proposal.

SBSI19, May 2024, 2019, Aracaju, Brazil
 ACM ISBN 978-1-4503-7237-4
<https://doi.org/10.1145/3330204.3330255>

CCS CONCEPTS

• Software and its engineering → Software creation and management → Software development process management → Software development methods

KEYWORDS

Software maintenance; Agile Methods; Kanban; Lean.

ACM Reference format:

Gibeon Soares de Aquino Junior, André Medeiros Dantas.2019. An Agile Approach Applied to Intense Maintenance Projects. In *SBSI19: XV Brazilian Symposium on Information Systems, May 2024, 2019, Aracaju, Brazil. ACM, New York, NY, USA, 9 pages*. https://doi.org/10.1145/3330204.3330255

1 Introdução

Sistemas de Informação de grande porte precisam se manter relevantes por décadas, tanto para justificar os altos investimentos de desenvolvimento quanto para continuar cumprindo a sua missão e dar suporte às mudanças inevitáveis do negócio envolvido. Tais mudanças podem ser tanto aperfeiçoamentos quanto correção de defeitos. Certos problemas ou novas necessidades que só são encontrados quando milhares de usuários começam a utilizar as funcionalidades das formas mais diversificadas encontrando irregularidades de diferentes naturezas, difíceis de serem identificadas durante o processo de construção. Desta forma, manter sistemas de grande porte é uma atividade complexa que exige agilidade e habilidade dos profissionais responsáveis.

Esse trabalho tem o objetivo de descrever uma abordagem para lidar com projetos de software que são submetidos a constantes solicitações de mudanças e requerem uma rápida resposta. A proposta é baseada na aplicação de técnicas e conceitos encontrados em alguns dos métodos ágeis mais populares, tais como SCRUM [14], Kanban [1] e Lean Software Development [12] com o objetivo de otimizar e aprimorar o processo de manutenção de sistemas envolvidos nesse cenário. Além de conceber essa abordagem, também foi possível avaliar a sua adoção em um projeto real de grande porte. Os resultados alcançados foram expressivos e sinalizam um potencial para aplicação em outros projetos de software dessa categoria.

Esse artigo está organizado da seguinte maneira: na Seção 2 é apresentado o contexto onde este tipo de abordagem se faz necessário. Na Seção 3, a abordagem proposta é apresentada em detalhes como os princípios e práticas a serem aplicadas. Na Seção

4 é demonstrada a implantação da abordagem com um estudo de caso real e seus resultados obtidos. Finalmente, na Seção 5 são apresentados os trabalhos relacionados e na Seção 6 são apresenta-se as considerações finais.

2 Contexto

No tempo de vida útil de sistemas de informação de grande porte bem-sucedidos espera-se que o seu tempo de vida seja muito maior que o tempo de desenvolvimento. Sistemas que podem ser utilizados de 10 a 30 anos normalmente passam dois terços desse tempo sendo mantidos por suas equipes de TI [13], e nesse período diversos tipos de mudanças precisam ser realizadas. De acordo com teoria unificada para evolução do software [10], sistemas desse porte e intensamente submetidos a diversos tipos de mudanças inevitavelmente serão regidos segundo as Leis de Lehman, como por exemplo: (a) a lei de crescimento contínuo, a qual indica que as funcionalidades desse sistema devem ser continuamente ampliadas para manter a satisfação do cliente; e (b) a lei da complexidade crescente, onde estabelece que a medida que o sistema evolui, necessariamente a complexidade aumenta associadamente.

As mudanças podem ser de diferentes naturezas, como correção de defeitos, incorporação de novas funcionalidades e renovação tecnológica de ambientes. Essa é uma categorização de mudanças muito comum, e normalmente é reconhecida através de três classificações do processo de manutenção [17]: (a) corretiva, caracterizada pela correção de erros; (b) adaptativa, responsável pela adaptação e atualização do ambiente às tecnologias mais recentes; e (c) a perfectiva, na qual funcionalidades são aperfeiçoadas ou criadas para melhor atender às necessidades dos usuários. Esses diferentes tipos de manutenção mostram como gerenciar o processo de manutenção pode se tornar complexo, pois lidar com demandas com características distintas exige a adequação dos fluxos de trabalhos que permitam atender às necessidades específicas de cada um desses tipos.

As correções de defeitos associadas à manutenção corretiva precisam ser entregues com prazos de poucos dias, ou em alguns casos de poucas horas. Outros tipos de mudanças também podem requerer um curto tempo de resposta mesmo não se tratando de correções do sistema. Existem solicitações que assim como as correções, normalmente não consomem muito tempo ou esforço de desenvolvimento, mas por outro lado o atendimento àquela demanda pode significar um importante ganho ao usuário. Solicitações como pequenas melhorias pontuais em funcionalidades existentes, como adicionar um filtro a uma busca ou enviar notificações por e-mail após a execução de alguma operação podem agregar mais valor ao sistema sem necessariamente consumir um grande esforço da equipe.

Da mesma forma outros tipos de solicitações também se encaixam com o mesmo perfil: criação de simples relatórios gerenciais que auxiliam no processo de tomada de decisão de gestores, exportação de dados brutos de necessidade específica extraídos diretamente da base de dados para também ajudar na gestão dos processos negociais, e ainda requisição de auditoria para investigar o comportamento do sistema especificamente em alguns momentos de uso, como por exemplo se um usuário realmente executou uma operação específica. Todos esses tipos de demanda requerem um fluxo contínuo de atendimento e um alto grau de colaboração entre as diferentes áreas do processo para que as solicitações dos usuários sejam atendidas e respondidas com eficiência e agilidade. Para esse fluxo deve-se considerar mudanças que exijam pouco esforço de desenvolvimento e impedir grandes demandas que onerem e criem gargalos no ritmo de entregas.

Mudanças perfectivas representam alterações mais significativas como na adaptação de interfaces para o usuário e regras de negócio, ou a criação de novas funcionalidades ou módulos. Demandas desse tipo normalmente requerem tempo e esforço maior de desenvolvimento, tanto na codificação quanto na validação e verificação. Considerando ainda o impacto mais significativo nos processos de negócio, essas alterações precisam ser gerenciadas em um fluxo planejado e com envolvimento direto das disciplinas mais fundamentais da engenharia de software como engenharia de requisitos, projeto, implementação e testes. Deve-se contar ainda com a participação do próprio cliente, ou usuários, por todo processo buscando garantir a conformidade entre o requisito e a implementação da mudança.

A complexidade desse cenário está inerentemente associada ao tamanho e impacto do respectivo software na comunidade de usuários em que está inserido. Sistemas de informação de grande porte proporcionam às equipes de desenvolvimento desafios diferentes em comparação a outros sistemas menores. Elas precisam estar preparadas para lidar com o fato de que a informatização de centenas de fluxos de negócio vai consequentemente motivar milhares de solicitações de mudanças no produto.

Outro fator influente, tão importante quanto a complexidade e tamanho do sistema, é o gerenciamento dos interesses e necessidades dos gestores de diversas áreas. Cada um deles com solicitações que abrangem desde mudanças para cumprir normas ou leis até melhorias apenas para agregar mais valor ao processo negocial. A transparência e confiança devem ser aspectos chave presentes no processo, com o objetivo de permitir uma boa execução do trabalho.

O processo de manutenção de sistemas de informação de grande porte precisa considerar todas essas variáveis permitindo uma forma eficaz e eficiente de atender o grande volume de solicitações de mudanças a que é submetido. No momento em que se qualifica e agrupa as mudanças pelas suas características, torna-se claro que a solução não pode resumir-se a um único tipo de tratamento no atendimento das mesmas. Enquanto demandas de caráter corretivo precisam ser tratadas de maneira rápida e contínua, as perfectivas devem ser planejadas e submetidas a um fluxo de trabalho com a participação do usuário e que permita aperfeiçoamentos mais robustos no sistema.

Este é justamente o contexto, bastante comum na manutenção de sistemas de informação de grande porte, no qual a nossa abordagem se aplica. Nestes ambientes há sempre um frequente conflito entre atender rápido demandas simples e ao mesmo tempo ter capacidade de realizar mudanças complexas em um ritmo constante. A metodologia proposta neste trabalho busca justamente endereçar simultaneamente estas duas facetas da manutenção de software, de forma articulada e sinérgica.

3 Metodologia

Este trabalho foi desenvolvido inteiramente em um ambiente real de manutenção de um sistema de informação de grande porte. Ele fez parte de uma iniciativa financiada pela própria instituição, com objetivo de fomentar, articular e executar ações de inovação, envolvendo o estudo, desenvolvimento e implantação de soluções inovadoras para melhoria da qualidade dos sistemas e aumento da eficiência das equipes envolvidas na sua manutenção. Neste sentido, as ações deveriam ser fundamentadas em avanços de pesquisas na área, mas deveriam ter resultados práticos e visíveis.

O foco desta pesquisa foi estudar, desenvolver, implantar e avaliar modelos de desenvolvimento de software, inspirados nas metodologias contemporâneas, com a finalidade de promover uma maior qualidade e eficiência no processo de manutenção em sistemas de grande porte e que são submetidos a um alto volume de demandas.

Além disso, ele ocorreu em um ambiente onde a teoria e prática coexistem de maneira sinérgica. Os desafios que motivam este trabalho de pesquisa são bem evidenciados somente em um contexto industrial. Ao mesmo tempo, o novo conhecimento é produzido como resultado de pesquisas e reflexões, e então, avaliados de volta no próprio ambiente prático. Por fim, os resultados desta aplicação servem como subsídios para ajustes nas teorias e práticas propostas, dando início a um novo ciclo. Este tipo de metodologia de pesquisa é conhecido como "Pesquisa-Ação" [5], que de acordo com Baburoglu e Ravn [2] tem como propósito resolver um problema real enquanto expande o conhecimento científico. Vezzosi [21] afirma que a Pesquisa-Ação não é um método linear, mas sim uma ação combinada entre prática, reflexão e aprendizado, cuja principal característica é ser de natureza cíclica e recursiva. De forma concreta, dois macroprocessos são executados de maneira cíclica e recursiva ao longo da pesquisa, são eles: **Pesquisa**: Revisão da literatura e propostas de abordagens com base em adaptações dos conhecimentos existentes ou mesmo o estabelecimento de novas teorias, técnicas e padrões. **Ação**: As propostas desenvolvidas são aplicadas ao contexto estudado, para poderem ser validadas, e as evidências sobre a eficácia das mesmas são coletadas e avaliadas.

4 Abordagem Proposta

Diante do contexto descrito na Seção 2, este trabalho propõe uma abordagem de manutenção de software capaz de lidar com o intenso fluxo de demandas de manutenção corretiva, de forma eficiente e eficaz, ao mesmo tempo que endereça a constante necessidade de evolução funcional e tecnológica de um sistema de grande porte. Se por um lado, as manutenções corretivas são essenciais para manter o sistema operante para os usuários, as evoluções são extremamente importantes para manter o sistema relevante e útil no longo prazo. O fato é que um sistema de grande porte nunca é completo e sempre necessita de evoluções [20].

Com o propósito de atender esses objetivos, a abordagem denominada **IMPRESS** - **I**ntense **M**aintenance **PR**oc**ESS** foi desenvolvida. Ela se baseia nos princípios e conceitos estabelecidos pelas áreas de conhecimento de Manutenção e Evolução de Software [8][9][11], ao mesmo tempo que incorpora práticas de processos bem difundidos no campo do desenvolvimento de software. Particularmente, ela absorve de maneira sinérgica vários dos princípios e práticas das metodologias ágeis, tais como SCRUM [14], Lean Sofware Development [12] e Kanban [1]. A filosofia adotada pelo IMPRESS é considerar que tipos de demandas diferentes devem ser tratadas através de estratégias diferentes, de modo a respeitar os seus requisitos particulares de tempo de atendimento, *stakeholders* envolvidos e, principalmente, complexidade e tamanho da solicitação. Desta forma, os diferentes tipos de demandas são organizados em duas principais categorias: Sustentação e Evolução.

A Figura 1 apresenta essa organização. Além disso, cada uma destas categorias de demandas é atendida por diferentes subequipes que trabalham com diferentes processos de desenvolvimento. Vale salientar que apesar de haver essa separação em subequipes, estas trabalham como um único time e dependendo do volume de demandas de cada uma das subequipes os membros de uma delas podem ser alocados na outra temporariamente.

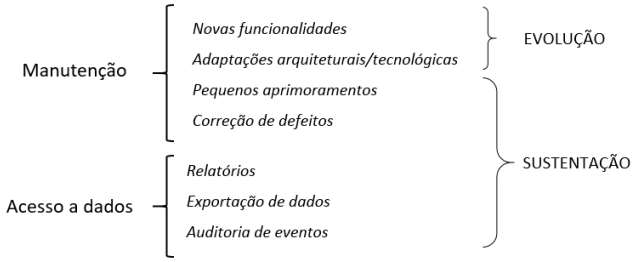

Figura 1. Categorização de demandas de manutenção adotadas neste trabalho

Além disso, todos trabalham na manutenção de um mesmo sistema e consequentemente em um mesmo repositório de código fonte. Por este motivo, a solução de processo precisa manter a equipe como um todo coesa e os dois processos precisam funcionar de maneira sincronizada e sinérgica. Para isto é recomendável que as duas subequipes sejam coordenadas pelo mesmo gerente de projeto, que os analistas de requisitos e arquitetos atuem simultaneamente em ambas e que estejam localizados no mesmo ambiente de trabalho.

A estrutura geral da solução proposta pode ser vista na Figura 2. Sistemas de grande porte com um grande número de usuários podem receber demandas de mudanças vindas de várias fontes, envolvendo diferentes grupos na instituição ou mesmo em outras instituições que também usam o sistema. Esse grande número de solicitações envolve os diversos tipos apresentados na Figura 1, normalmente sem qualquer categorização prévia ou indicação de criticidade. Desta forma, é necessário que se faça uma triagem a fim de se avaliar a coerência do pedido e categorizar com mais precisão o tipo de demanda e o nível de urgência da mesma. Esse trabalho deve ser feito por um grupo com características similares a um CCB (*Change Control Board*), que possui um bom conhecimento do negócio da instituição, das funcionalidades e do plano de evolução do sistema e, finalmente, dos aspectos técnicos e arquiteturais da solução. Normalmente, fazem parte deste grupo representantes que envolvem analistas de negócio e requisitos, gerente do projeto e arquiteto [7]. Com base no resultado desta triagem as demandas são

direcionadas para o *backlog* de tarefas de cada uma das subequipes: Sustentação e Evolução.

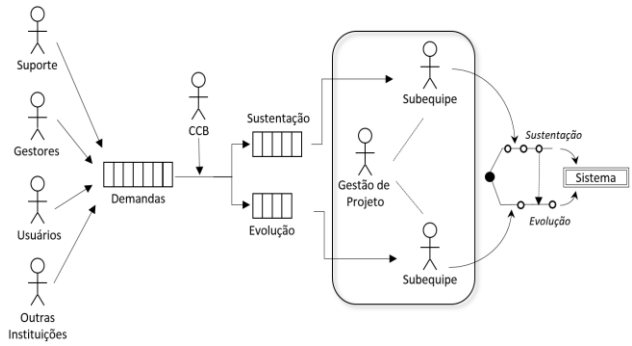

Figura 2. Estrutura geral da abordagem IMPRESS

O IMPRESS é uma abordagem de processo que une dois dos principais métodos ágeis comumente usados em desenvolvimento de software: SCRUM e Kanban, com um propósito muito específico de manutenção e evolução de sistemas de grande porte que possuem um intenso fluxo de mudança. Como tais métodos são amplamente conhecidos pela comunidade de acadêmica e da indústria, neste artigo serão detalhadas somente as práticas mais gerais que não fazem parte do escopo de cada um dos métodos.

No fluxo de sustentação, as tarefas são analisadas pela subequipe responsável assim que chegam e colocadas em uma fila de prioridades. O atendimento às mesmas faz o uso de estratégia inspirada nas técnicas comumente usadas em sistemas operacionais, tais como escalonamento por prioridade e escalonamento por prazo [19], de modo a garantir que o atendimento mais ágil às demandas mais críticas, ao mesmo tempo que evita a ocorrência do problema da inanição. A sustentação é fortemente baseada no método Kanban, que devido às suas características de ser uma abordagem que permite a entrega *just-in-time*, ter um fluxo de trabalho incremental e evolucionário, além de focar na visibilidade do andamento das tarefas e colaboração do time, se mostra uma ferramenta ideal para o atendimento às demandas categorizadas neste tipo. Uma grande vantagem do Kanban é que ele não impõe qualquer atividade específica, permitindo assim que possa ser implantado e adequado ao atual fluxo de atividades da organização.

O funcionamento do subprocesso do IMPRESS, referente à sustentação do sistema, se baseia nos seguintes pilares: (a) Entregar valor (demanda resolvida em produção) continuamente e o mais rápido possível, através do uso de técnicas de priorização de tarefas; (b) Visibilidade e controle do fluxo de tarefas através do quadro Kanban e suas práticas; (c) Engajamento ponta-a-ponta através da participação de todos os envolvidos no ciclo de vida das tarefas; (d) Monitoramento contínuo através de estabelecimento de metas de curto prazo (semanais), reuniões diárias e manutenção em tempo real de métricas e *dashboards* públicos.

Já o subprocesso do IMPRESS relativo ao fluxo de Evolução do sistema é concebido com base nas práticas já bem difundidas de metodologias ágeis, em particular, as práticas do SCRUM. Neste, o *backlog* de tarefas é mantido e alimentado constantemente a partir das solicitações que são categorizadas como do tipo evolutiva. Normalmente, sistemas de grande porte abrangem diversas áreas de uma instituição e cada uma dessas requer evoluções diferentes nos módulos relacionados às suas áreas de atuação. A grande dificuldade de uma área de TI em um contexto deste é saber como priorizar o desenvolvimento das tarefas de modo a maximizar o valor para a instituição como um todo. O SCRUM preconiza a figura do *Product Owner*, que tem o papel central de determinar essa prioridade. No entanto, em ambientes institucionais com um sistema de grande porte é impraticável determinar uma única pessoal para atuar com este papel. Considerando esse problema, a abordagem do IMPRESS recomenda que seja designado um comitê de negócios, envolvendo representantes das diferentes áreas da instituição e que tenham uma boa visão dos processos de negócio, de modo a poder influenciar a tomada de decisões sobre as prioridades a serem desenvolvidas. Com esta adaptação, é possível ter um *backlog* de tarefas sempre priorizado, de acordo com a visão de valor para os *stakeholders*, de modo a permitir a seleção das tarefas mais importantes para o escopo de cada *sprint*. As demais práticas do SCRUM se adequam perfeitamente às demandas direcionadas ao fluxo de sustentação. Vale salientar que o IMPRESS absorve toda a filosofia do SCRUM para esses tipos de demandas e por isto os princípios e práticas originais deste método ágil são respeitados.

Por fim, os resultados das manutenções e evoluções no sistema executados por duas equipes distintas, com processos diferentes e ciclos de desenvolvimento com tempos diferentes precisam ser integrados em um único ponto e muito bem controlado já que modificam o mesmo código fonte de um único sistema em produção. Para atender a esses requisitos padrões de gerência de configuração e mudança precisam ser bem aplicados. A abordagem IMPRESS além de incorporar alguns desses padrões largamente usados para esta finalidade, tais como, Release Line, Release-Prep Codeline, Active Development Line e Integration build [4], se inspira fortemente no modelo de gerência de configuração denominado Gitflow[1].

5 Implantação da Abordagem e Resultados Alcançados

A proposta do IMPRESS pode ser aplicada em um ambiente real de um projeto de software com características de um sistema de informação complexo de grande porte. Este estudo de caso foi realizado sob o projeto de desenvolvimento do sistema da UFRN chamado SIGAA. Nessa seção serão apresentados alguns aspectos importantes sobre o projeto, seguido pelas descrições da forma como foi realizada a implantação da abordagem para manutenção do sistema envolvendo os fluxos de Sustentação e Evolução. Por fim, serão apresentados alguns resultados coletados após três anos de uso da abordagem.

[1] http://nvie.com/posts/a-successful-git-branching-model/

5.1 O projeto de Manutenção do Sistema

Esse sistema, em uso desde 2007, é responsável pelo controle dos dados e a informatização dos processos institucionais sob os mais variados aspectos da instituição. Contando com 44 módulos, milhares de funcionalidades e aproximadamente 38.000 pontos de função, o sistema oferece operações que abrangem as principais áreas. Devido a esta amplitude, o sistema passou a ser a ferramenta indispensável para a execução de praticamente qualquer processo na instituição.

Além do tamanho do sistema, o projeto atende a uma comunidade de mais de 50 mil usuários da própria instituição entre docentes e discentes, que produzem uma demanda de acesso mensal (representada na Figura 3) na ordem de 10 milhões de visualizações de páginas, com um pico de quase 14 milhões em Julho de 2018. Este cenário produz, consequentemente, altos números de solicitações de mudanças no sistema. Nos últimos 3 anos, de Setembro/2015 a Setembro/2018 a equipe implementou aproximadamente 7.000 tarefas de mudanças no sistema.

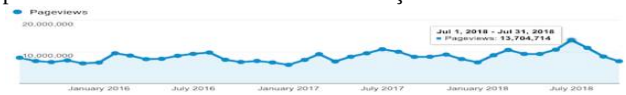

Figura 3. Gráfico de acesso diário às páginas do sistema

Este contexto apresenta enormes desafios, envolvendo àqueles de caráter de desenvolvimento de software, outros relacionados à garantia de alta disponibilidade do sistema, e também de caráter gerencial, exigindo habilidade de negociação para lidar com os interesses descentralizados de diversos gestores da instituição. Diante desse cenário o processo de manutenção de software do projeto passou a enfrentar problemas como longos períodos para entregar mudanças solicitadas e consequentemente acúmulo de solicitações sem atendimento. Situação fortemente influenciada por aspectos negativos do processo de desenvolvimento legado.

A implantação do IMPRESS no projeto do SIGAA buscou otimizar todo processo de manutenção do sistema. Os fluxos de Sustentação e Evolução foram implantados de acordo com a proposta descrita neste trabalho resultando em melhorias significativas, as quais serão discutidas ainda nessa seção. A seguir são descritos como esses fluxos foram implantados e quais resultados puderam ser obtidos.

5.2 Kanban em Sustentação

O uso do quadro Kanban para Sustentação é a essência da solução proposta, o qual proporciona a transparência e simplicidade necessária ao processo. O desafio desse fluxo de trabalho é conseguir gerenciar as demandas que chegam constantemente todos os dias, com diferentes graus de criticidade.

A montagem do quadro foi feita diretamente na parede utilizando papéis adesivos para representar cada solicitação de mudança ou tarefa de desenvolvimento (como mostrado na Figura 4). A escolha pelo uso do quadro físico, em contraste ao uso de aplicativos virtuais, deu-se com o objetivo de promover ainda mais a visibilidade e facilitar as reuniões diárias entre todos os envolvidos.

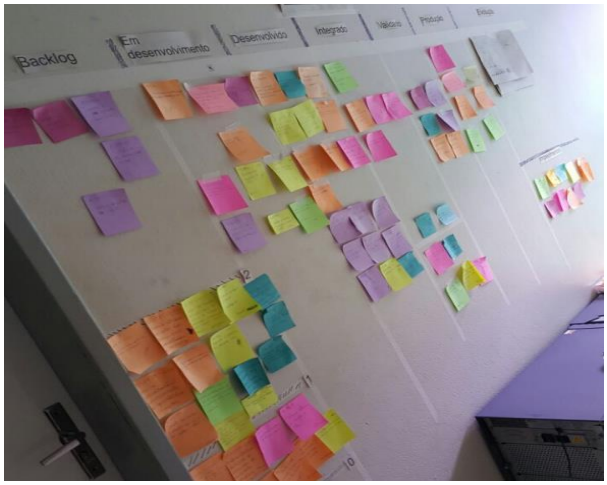

Figura 4. Quadro Kanban montado na parede do corredor do ambiente de trabalho da equipe de desenvolvimento

Cada seção do quadro corresponde a uma etapa no processo de resolução demanda, como pode ser observado na Figura 5. Outro aspecto visual importante é a diferenciação das cores de cada tarefa. Por exemplo, os papéis rosas representam tarefas de mais alta criticidade – referentes à problemas que estão bloqueando a realização de uma dada funcionalidade para um alto número de usuários. As demais cores: azul, amarela e laranja, indicam demandas que chegam diariamente, e denotando respectivamente a ordem crescente de prioridade de atendimento.

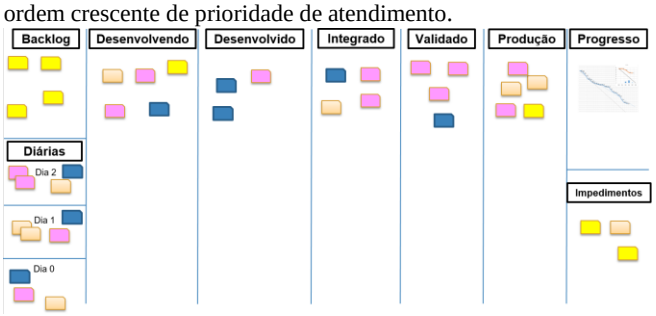

Figura 5. Quadro Kanban adaptado para o processo de Sustentação

Como mencionado anteriormente, a maior parte das seções do quadro estão diretamente relacionadas às etapas técnicas do processo de desenvolvimento como implementação e testes, porém existem algumas seções com propósitos mais específicos. As seções do quadro foram divididas da seguinte forma: (a) a primeira coluna, dedicada a mostrar as solicitações pendentes que devem entrar no processo de desenvolvimento, está dividida em duas partes: um backlog finito de demandas simples de baixa prioridade herdadas do processo de desenvolvimento legado, e as demandas que chegam diariamente; (b) as etapas internas do processo de desenvolvimento passam a ser evidenciadas na transição das demandas entre as colunas “Desenvolvendo” e “Produção”; e finalmente, (c) na última coluna são apresentadas duas seções com objetivos distintos, uma para exibir dados estatísticos sobre o progresso de produtividade da

equipe e outra para agrupar e destacar tarefas que se encontram – impedidas de serem resolvidas enquanto algum obstáculo não for resolvido.

Em algumas dessas seções, seguindo as recomendações do uso do Kanban, se faz necessário aplicar limites máximos de tarefas. Essas restrições devem ser aplicadas nas etapas "Desenvolvendo" e "Integrado". A primeira representa o estágio inicial quando um desenvolvedor escolhe uma tarefa para resolver e então move o papel para essa seção com seu nome marcado, e a segunda representa o momento em que um testador indica qual tarefa ele começará a validar. O limite máximo em ambas etapas deve ser o número total de pessoas que participam desse fluxo de trabalho exercendo aquele papel, ou seja, nunca deve haver um desenvolvedor ou testador com mais de uma tarefa associada ao seu nome nas respectivas seções. Essa medida busca evitar o desperdício e garantir o foco necessário para entregar o que já começou a ser processado pelo fluxo.

Outra prática recomendada na aplicação do subprocesso de Sustentação, e que também auxilia na minimização do desperdício, é entrega contínua acontecendo com frequências diárias, ou em alguns casos, mais de uma vez por dia. Com isso o processo naturalmente leva o time a manter sua produção regular e sustentável, evitando também gargalos no fluxo.

5.2.1 Resultados de Sustentação. A proposta do fluxo de Sustentação foi implantada no processo de manutenção do SIGAA a partir do mês de setembro do ano de 2015. Antes dessa implantação o atendimento às mudanças não era realizado de forma sistemática e organizada, o que resultou em quadros de insatisfação da comunidade usuária do sistema e uma falta de credibilidade na capacidade da equipe de resolver as solicitações. Com a implantação do processo foi possível perceber melhoras significativas na produtividade da equipe e uma maior confiança dos usuários em registrar solicitações de mudança e informar os problemas do sistema.

A Figura 6 demonstra essa melhoria a partir da análise de duas métricas, calculadas por semestre, são elas: (a) O tempo médio para finalizar a tarefa – representa o tempo total em dias passados desde o momento que o usuário reporta o problema até o momento que a correção é efetivamente colocada em produção; e (b) média do número de tarefas criadas por dia – contagem da média de solicitações reportadas por dia. A introdução de uma nova dinâmica para atender as demandas não interferiu diretamente na forma com que as tarefas eram gerenciadas, e por esse motivo foi possível comparar essas métricas citadas antes e depois da implantação do fluxo aqui proposto.

Como pode ser visto no gráfico antes da implantação do IMPRESS o tempo médio de finalização das demandas era de 25.5 dias e no mesmo semestre da implantação já foi possível perceber uma diminuição de 5.2 dias (aproximadamente 20%). Além disso, é possível perceber que o impacto foi consistente, permitindo uma melhoria contínua dos números nos períodos posteriores. A última medição (2018.1), três anos após a implantação, demonstra o resultado sólido e expressivo de melhoria (redução de mais de 50%) nesta métrica.

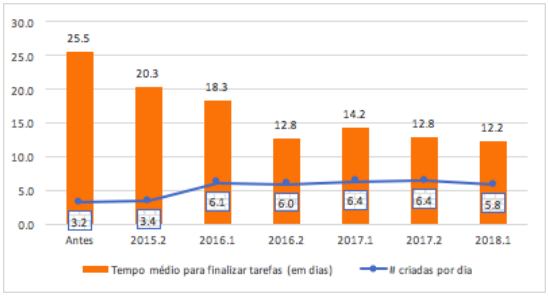

Figura 6. Comparação entre a média de tarefas criadas por dia e o tempo médio de atendimento.

Outro resultado interessante, que é possível perceber no gráfico, é que houve um aumento no número de solicitações registradas pelos usuários. Não é possível afirmar com precisão a causa deste efeito, mais a hipótese é que este aumento se deu em decorrência do aumento da confiança que os usuários passaram a ter sob a capacidade da equipe em efetivamente atender às suas solicitações em um tempo razoável. Em decorrência disto os usuários passaram a reportar com mais frequência os problemas encontrados, e principalmente, a reportar sugestões de melhorias nas funcionalidades do sistema. Um outro aspecto a observar nos resultados destas duas métricas é que mesmo com o aumento significativo no número de demandas, a equipe foi capaz de diminuir o tempo médio para resolução das mesmas.

No que diz respeito aos números absolutos de tarefas entregues após a implantação do IMPRESS a Figura 7 apresenta um gráfico demonstrando o crescimento regular do total de atendimentos. O que confirma que a melhoria no processo teve um efeito permanente na capacidade da equipe e também na confiança por parte dos requisitantes.

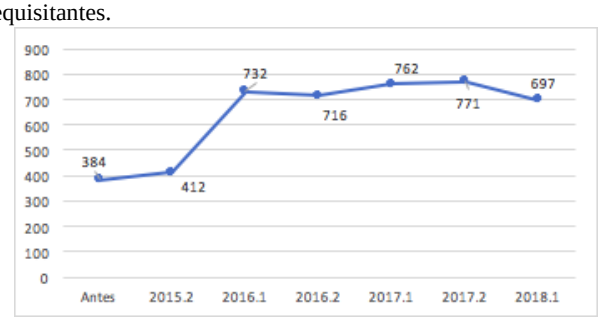

Figura 7. Totais de tarefas entregues em sustentação desde a implantação do IMPRESS

Essas métricas são fundamentais para perceber que o aumento no número de entregas foi acompanhado e justificado pela otimização do processo, além de refletir um avanço na interação e colaboração entre os times de desenvolvimento e de suporte ao usuário. Todos os resultados indicam melhorias significativas em aspectos importantes do processo, como por exemplo o aumento de produtividade ao atender mais demandas em metade do tempo e ainda a diminuição de desperdícios em estágios cruciais do fluxo. Além desses fatores analisados, a partir das métricas extraídas do processo, a nova forma de trabalhar propiciou outros benefícios que

não puderam ser capturados com números. As mudanças do processo impactaram na forma como os times passaram a colaborar e trocar informações entre si, além de propiciar, talvez a vantagem mais direta e óbvia do Kanban, uma maior visibilidade do trabalho realizado.

5.3 Scrum em Evolução

A implantação do SCRUM em qualquer projeto requer pelo menos um elemento básico e imprescindível, um backlog de demandas priorizadas. O backlog de demandas de evolução para sistemas de grande porte é um dos poucos artefatos que nunca será completamente finalizado. Seguindo justamente uma das leis de Lehman na qual prescreve que as melhorias e aprimoramentos de funcionalidades do sistema são contínuas para o sistema nunca cair em desuso. Essa lista de demandas é um instrumento "vivo" e dinâmico que precisa estar sempre sendo priorizado para atender continuamente aos interesses dos usuários.

O principal desafio dessa implantação foi lidar com a definição do papel do *Product Owner* (PO), que é essencial para adoção das práticas estabelecidas no SCRUM. Devido à pluralidade de áreas afetadas pelo sistema atual, e especialmente a autonomia de suas gestões, é inviável definir um representante desse papel para responder por todas as áreas envolvidas.

A ausência desta figura no processo de evolução dos sistemas já tinha sido apontada como um grande problema em um estudo anterior. Particularmente, o estudo demonstrava que ao longo dos anos houve um grande desperdício de recursos, devido a implementação de funcionalidades que ao serem finalizadas eram pouco utilizadas ou, em alguns casos, eram completamente abandonadas. O estudo apontou que este problema era gerado, principalmente, pela incapacidade de se priorizar as demandas com base na importância estratégica da mesma para a instituição.

Diante desta problemática, foi definido que uma parte do papel do PO, particularmente a priorização das demandas, seria realizado por um comitê, composto por representantes das principais áreas envolvidas no sistema. Este comitê foi nomeado como **CoPES** – **Comitê para Priorização da Evolução dos Sistemas**. Para este comitê foram nomeados 5 membros, que representam todas as áreas da instituição. O comitê se reune periodicamente para estabelecer as prioridades das demandas de evolução do sistema e assim manter um *backlog* de demandas devidamente priorizadas. A priorização é feita considerando-se a relação entre o custo e benefício de cada demanda.

Adicionalmente, foi estabelecido que um conjunto de câmaras técnicas dariam o suporte ao trabalho do comitê. Basicamente, estas câmaras deveriam realizar a coleta das informações técnicas necessárias para uma melhor compreensão da demanda, de modo que o comitê se concentre na avaliação estratégica das demandas. Foram estabelecidas sete câmaras, cobrindo todas as áreas de negócio da instituição, são elas: **Gestão Financeira** – Grupo responsável por demandas relativas à distribuição de recursos financeiros pelas unidades, liquidações e pagamentos de contratos e convênios, etc., relacionando as contas de cada projeto às rubricas contábeis, das fundações e estas com as do Governo Federal; **Gestão de Aquisições e Contratações** – Grupo responsável por demandas relativas à compras, estoques e patrimônio, além da logística de distribuição dos recursos físicos e transporte entre as unidades; **Gestão de Infraestrutura** – Grupo responsável por demandas relativas à acompanhamento e controle das obras, sejam de novos prédios, bem como a manutenção, administração da infraestrutura física da instituição já existente e distribuída em vários campi; **Gestão da Informação** – Grupo responsável por demandas relativas à comunicações oficiais e processos, envolvendo memorandos, processos, ofícios, portarias, boletim de serviços, etc.; **Gestão de Pessoas** – Grupo responsável por demandas relativas à gestão da alocação das pessoas nas unidades, da projeção e distribuição de vagas, avaliação do desempenho e ajustes através de capacitação ou recolocação de pessoas em postos com maior afinidade, segurança e saúde, entre outros aspectos envolvendo atendimento de pessoas; **Serviços Corporativos** – Grupo responsável por demandas relativas aos aspectos da gestão administrativa, ambiental, de qualidade, e de apoio aos alunos (RU, casas de estudante, bolsas, etc.); **Controle e Auditoria** – Grupo responsável por demandas relativas ao apoio aos órgãos de controle e assessoria como a auditoria e a assessoria jurídica que auxiliam à administração a se manter em conformidade com a legislação do governo federal.

A partir do estabelecimento de um *backlog* válido foi possível implantar o fluxo de evolução proposto no IMPRESS. Todas as *sprints* são executadas usando as principais atividades prescritas pelo SCRUM como: (a) reunião de planejamento com todo time para estimar os tamanhos dos itens da sprint usando a técnica do *planning poker* [6]; (b) a discriminação das subtarefas que serão acompanhadas no quadro de tarefas; e (c) a reunião de retrospectiva para a avaliação e diagnósticos dos problemas e conquistas de cada sprint. Durante a execução das sprints de 20 dias úteis (ou um mês) existe ainda o acompanhamento de um gráfico de *burndown* com as horas consumidas que foram planejadas para cada subtarefa, e com isso dar visibilidade a todo time do progresso do ciclo.

5.3.1 Resultados de Evolução. Um dos resultados mais concretos que surgiram após a implantação do IMPRESS no fluxo de Evolução foi o advento do *backlog* como artefato institucionalizado e confiável. O uso do *backlog* trouxe ainda outro benefício à realidade do projeto, além do seu surgimento como artefato. Após a introdução da prática de mantê-lo priorizado de acordo com as necessidades dos gestores, a negociação com os mesmos se tornou muito mais transparente. Em alguns casos acabou ocorrendo a consequência natural do próprio gestor reconhecer o volume de demandas com alta prioridades já solicitadas, e reavaliar o escopo da mesma para se adequar melhor à relação entre custo e benefício.

Diferentemente do que aconteceu com a implantação da abordagem em Sustentação, o impacto sentido no projeto foi mais difícil de mensurar objetivamente. Entretanto esses impactos, mesmos que subjetivos, já indicam importante avanços com a transparência e a priorização do que de fato é relevante ao usuário.

6 Trabalhos Relacionados

Fehlmann e Falkner [15] realizaram um estudo exploratório do uso do SCRUM em um projeto de manutenção de um software no Australian Defence Science and Technology Organisation (DSTO). Os autores reportam resultados positivos na adoção da metodologia.

No entanto, o contexto do projeto não se caracteriza como de manutenção intensa, pois considera ciclos de manutenção com duração de duas a quatro semanas. O IMPRESS se diferencia desta solução por permitir simultaneamente ciclos curtos (diários) para tarefas urgentes e ciclo longos para manutenções caracterizadas como evolução.

Renuka et al [16] também realizaram um estudo de caso do uso do SCRUM em um projeto de evolução de software envolvendo uma equipe de aproximadamente 60 pessoas. Apesar do projeto ser de alta complexidade, ele também não se caracteriza como de manutenção intensa, pois adota ciclos de manutenção com duração de duas a quatro semanas.

No trabalho de Harald e Höst [18] é descrito a experiência de implantação do método eXtreme Programing - XP [3] em uma organização de desenvolvimento de software de grande porte durante oito meses. Eles reportam que a introdução deste método da maneira original é bastante difícil e por isso diversas adaptações precisaram ser feitas para adotá-lo. Este trabalho se diferencia bastante do nosso porque o IMPRESS não recomenda a metodologia XP, apesar de ser possível usá-la para as práticas de engenharia. A ênfase da nossa abordagem está no controle dos fluxos de demandas e gestão ágil do projeto.

7 Considerações Finais

Um dos maiores desafios do gerenciamento do processo de manutenção de sistemas complexos de grande porte é controlar o grande volume de solicitações de mudanças e conseguir dar um retorno ao grande número de usuários em tempos aceitáveis. A proposta apresentada neste trabalho pode auxiliar a minimizar essa complexidade. Através da implantação do IMPRESS, no qual são identificados fluxos de trabalho distintos de acordo com a natureza da mudança necessária ao sistema, nomeados aqui como Evolução e Sustentação, é possível manter um ritmo de trabalho adequado e atender aos diferentes propósitos da manutenção. No primeiro são tratadas mudanças solicitadas que precisam de um ciclo de desenvolvimento mais longo para análises mais aprofundadas por se tratarem de alterações mais impactantes e custosas. No segundo, o fluxo requer, em contraste com as sprints mensais de evolução, entregas diárias de demandas que não requerem esforço ou tempo demasiado de desenvolvimento, mas que precisam ser entregues ao usuário com eficácia e eficiência.

A implantação dessa proposta foi muito bem sucedida para o projeto de manutenção do sistema SIGAA. Em pouco tempo foi possível observar resultados relevantes, que em comparação a números de períodos anteriores, indicaram um impacto muito positivo na produtividade e eficiência na entrega das mudanças solicitadas. Além de proporcionar importantes desafios, inerentes à natureza da solução, que surgiram como restrições a serem consideradas em possíveis adesões a essa abordagem, tais como: gestão de subequipes para paralelizar o desenvolvimento, o controle da classificação do tamanho e prioridade de demandas de sustentação para evitar que tarefas grandes ou complexas não criem gargalos significativos, e ainda envolvimento efetivo da alta gestão no comitê de priorização das demandas de evolução.

Apesar da proposta ter sido validada em um único projeto, ela pode ser aplicada para outros tipos de projetos de manutenção de sistemas de informação de grande porte. As práticas adotadas e os princípios por trás do IMPRESS representam uma compilação sinérgica das melhores práticas de desenvolvimento e manutenção de software. Desta forma, acreditamos que outros projetos de natureza similar possam também obter resultados positivos com adoção desta abordagem.

AGRADECIMENTOS

Os autores agradecem à Superintendência de Informática da UFRN pela oportunidade de implantar a abordagem proposta neste trabalho no projeto de desenvolvimento do SIGAA.

REFERÊNCIAS

[1] Anderson, D. 2010. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press.

[2] Baburoglu, O. N. and Ravn, I. (1992). Normative Action Research. Organization Studies, 13(1), 019–34.

[3] Beck, Kent. Extreme programming explained: embrace change. addison-wesley professional, 2000

[4] Berczuk, Stephen P., e Brad Appleton. 2002. *Software configuration management patterns: effective teamwork, practical integration*. Addison-Wesley Longman Publishing Co., Inc.

[5] Brydon-Miller, M., Greenwood, D., and Maguire, P. (2003). Why Action Research? Action Research, 1(1), 9–28.

[6] Cohn, M. 2005. *Agile Estimating and Planning*. Prentice Hall.

[7] Hass, Glen. 2003 *Configuration management principles and practice*. Addison-Wesley Longman Publishing Co., Inc.

[8] JTC, ISO. 2006. "1/SC 7." Software Engineering–Software Life Cycle Processes–Maintenance," ISO/IEC 14764 (2006).

[9] Lehman, M. 1979. *On understanding laws, evolution, and conservation in the large-program life cycle*. Journal of Systems and Software 1 (1979): 213-221.

[10] Lehman, M. e Belady, L. 1997. *Program Evolution: Processes of Software Change*. Academic Press

[11] Lientz B., Swanson E., 1980: *Software Maintenance Management*. Addison Wesley, Reading, MA

[12] Poppendieck, M. e Poppendieck, T. 2013. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional.

[13] Pressman, R. 2014. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.

[14] Schwaber K. e Sutherland J. 2010. *The SCRUM Guide*. SCRUM.org.

[15] Shannon Fehlmann and Katrina Falkner. 2015. A case study in agility and evolving the long-lived software system. In Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference.

[16] Sindhgatta, Renuka, Nanjangud C. Narendra, and Bikram Sengupta. "Software evolution in agile development: a case study." Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion. ACM, 2010.

[17] Sommerville, I. 2010. *Software Engineering*. Addison Wesley.

[18] Svensson, Harald, and Martin Höst. "Introducing an agile process in a software maintenance and evolution organization." Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on. IEEE, 2005.

[19] Tanenbaum, Andrew S., and Herbert Bos. 2014. *Modern operating systems*. Prentice Hall Press.

[20] Václav Rajlich. 2014. *Software evolution and maintenance*. In Proceedings of the on Future of Software Engineering (FOSE 2014). ACM, New York, NY, USA, 133-144.

[21] Vezzosi, M. (2006). Information literacy and action research. New Library World, 107(7/8), 286–301.