Os benefícios do uso de Kanban na gerência de projetos de manutenção de software

Diogo Vinícius de S. Silva, F. Alan de O. Santos, Pedro Santos Neto

Infoway Tecnologia e Gestão em Saúde LTDA
Rua Antônio Tito, 223 Bairro Jockey – Teresina – PI – Brasil.

{diogo,alansantos,pasn}@infoway-pi.com.br

Abstract. *In development teams dealing with legacy systems, work is generally governed by uncertainty and unpredictability. This peculiarity represents a challenge for the effective planning and monitoring of the project, since there in no way to avoid the emergence of unplanned demands. This paper presents a successful experience of using Kanban, together with other agile practices, for managing a legacy system developed at Infoway-PI. The difficulties and solutions found in this experience are reported.*

Resumo. *Em equipes de desenvolvimento que lidam com sistemas legados, geralmente o trabalho é regido pela incerteza e pela imprevisibilidade. Tal peculiaridade representa um desafio para o planejamento e acompanhamento efetivo do projeto, uma vez que não há como evitar o surgimento de demandas não planejadas. Este artigo apresenta uma experiência de sucesso do uso de Kanban, juntamente com práticas de outras metodologias ágeis, para o gerenciamento de um sistema em produção desenvolvido na Infoway-PI. As dificuldades e soluções encontradas nessa experiência são relatadas.*

1. Introdução

Assim como quase todos os tipos de produtos, softwares de sucesso evoluem constantemente com o decorrer do tempo. Tal evolução, também chamada de manutenção [IEEE 1998], é muito comum nos sistemas computacionais. A manutenção de software pode ser considerada um processo geral de modificação de um software, depois de ter sido colocado em uso, normalmente não envolvendo alteração da arquitetura do mesmo. A manutenção é normalmente dividida de acordo com o seu objetivo, podendo ser [ISO/IEC 14764 1999]:

- Corretiva: para reparar defeitos;
- Adaptativa: para adaptar o software a alguma mudança no seu ambiente de execução;
- Perfectiva: para melhorar algo já funcional, sem alteração no comportamento visível ao usuário.
- Preventiva: para corrigir falhas antes que aconteçam.

Os tipos de manutenção citados anteriormente englobam todos os tipos de mudanças necessárias durante o ciclo de vida de um software. Segundo Pigoski (1996) e Pfleeger (2001), a manutenção de um software corresponde à

parte mais onerosa do desenvolvimento, podendo representar até 90% do custo total envolvido. Apesar da importância da manutenção, o gerenciamento de projetos desta natureza ainda tem sido pouco explorado pelas metodologias atuais [MCT 2009].

Este artigo apresenta o conhecimento obtido na experiência da aplicação do método Kanban como ferramenta para melhoria da gestão de projetos de software em uma equipe de manutenção de sistemas, utilizando uma abordagem ágil baseada em Scrum, no contexto da Infoway Tecnologia e Gestão em Saúde LTDA. O relato se dá por meio do detalhamento das atividades realizadas pela equipe entre os meses de janeiro e maio de 2011, em uma iniciativa dos próprios membros da equipe em busca da melhoria contínua e da gestão e disseminação do conhecimento.

Este trabalho está organizado da seguinte forma: na Seção 2 é feita uma descrição do cenário da organização. Na Seção 3 são abordados os indicadores que motivaram o estudo e a adoção de práticas adicionais à utilização do Scrum tradicional. As principais características do método Kanban são expostas na Seção 4. As ações executadas no intuito de incorporar essas práticas na gestão de projetos existente são relatadas na Seção 5. Na Seção 6, são descritos os benefícios e dificuldades encontradas, além dos fatores de sucesso. A Seção 7 apresenta as limitações do trabalho. A Seção 8 discute sobre os trabalhos relacionados. Por fim, na Seção 9 são feitas as considerações finais.

2. O cenário organizacional

A Infoway é uma empresa privada focada no desenvolvimento de sistemas para gestão de planos de saúde. Há 15 anos no mercado, foi apenas há seis que se iniciou o uso de metodologias de desenvolvimento de softwares para a gerência dos projetos. Este início se deu devido ao crescimento da empresa e ao surgimento de projetos para serem desenvolvidos em paralelo por diferentes equipes, pois, neste caso, foi necessário um nível maior de organização no seu funcionamento.

A gerência de tecnologia da Infoway está dividia em coordenações. Cada uma é responsável por um projeto diferente sendo composta por um coordenador (gerente do projeto) e sua respectiva equipe (geralmente de três a seis desenvolvedores). Atualmente existem cinco projetos sendo desenvolvidos em paralelo na Infoway.

Cada coordenação tem autonomia suficiente para utilizar o processo que melhor se adeque às necessidades do contexto no qual estão inseridos. Devido à grande difusão do framework Scrum [Schwaber 2002] para o gerenciamento de projetos, todas as equipes optaram pelo uso deste framework para o gerenciamento das atividades, tendo o papel de Scrum Master desempenhado pelo coordenador do projeto. Aliado ao Scrum, também são aplicadas práticas XP (*eXtreme Programming*) como refatoração, programação em par, desenvolvimento orientado a testes, dentre outras [Beck 2004].

Com o intuito de buscar melhores soluções para o desenvolvimento de *softwares*, os membros de uma das equipes decidiram experimentar o uso do método Kanban [Anderson 2010]. Devido à natureza do software pelo qual a equipe é responsável (em produção e constantes melhorias) sentiu-se a necessidade de experimentar outros modos de gerenciamento do projeto. A equipe em questão foi a que

responde pelas demandas do projeto Uniplam. Este projeto é composto por quatro desenvolvedores e um coordenador, que são responsáveis por atender o cliente (gestores do plano de saúde Uniplam) que diariamente demandam:

- Relatórios gerenciais;
- Ajustes em módulos em produção;
- Novas funcionalidades;
- Correção de *bugs*.

3. Indicadores da necessidade de mudança

Depois de algum tempo aplicando Scrum, verificou-se que ainda eram necessários outros modelos para controle dos problemas que possam surgir em um ambiente de um sistema em produção.

O principal fator que contribuía para a não continuidade do fluxo normal de trabalho era a grande quantidade de demandas extras, ou seja, solicitações urgentes dos clientes que não estavam planejadas. Frequentemente isso acarretava na realocação de membros da equipe para atender a essas demandas durante a execução das tarefas da iteração, que no Scrum é denominado *sprint*.

Para amenizar o comprometimento do *sprint*, medidas paliativas eram tomadas. Uma dessas medidas foi incluir no planejamento uma quantidade de horas de trabalho que se destinassem às demandas extras. Essa quantidade de horas foi obtida com base no histórico do número médio do tempo de resolução das solicitações extras que surgiam durante uma iteração. Em alguns casos esse número chegou a representar 50% do tempo do *sprint*. Já em outros casos, o número de horas extras chegava a representar apenas 10% do tempo. Essa grande variação foi uma das grandes dificuldades encontradas para o gerenciamento dessas atividades. Numa tentativa de minimizar esse problema outra abordagem foi testada: incluir menos horas para os desenvolvedores no *sprint*. No entanto, o problema ainda continuava, pois ainda não era possível controlar as tarefas, visto que algumas vezes, já que as tarefas extras ainda continuavam sem previsão, o desenvolvedor iria trabalhar menos do que o programado para o *sprint.*

Outro problema encontrado nesta abordagem foi a impossibilidade de se realizar um acompanhamento efetivo para as tarefas extras. A real capacidade da equipe de atender essas solicitações e a priorização pelo cliente se tornava complicada, uma vez que a maioria das demandas chegava com prioridade alta. Sendo assim, a equipe acabava parando o desenvolvimento do que tinha sido planejado para dar uma resposta rápida ao cliente que, devido ao sistema estar em produção, não tinha como esperar uma próxima iteração para a equipe planejar e iniciar essas atividades. Na maioria dessas demandas extras era essencial a realização do trabalho operacional do cliente. Diante disso, as metas dos *sprints* continuavam não sendo atingidas.

Outra dificuldade observada com o uso do Scrum foi o acúmulo excessivo de tarefas em um determinado estado do fluxo de trabalho, por exemplo, o acúmulo de muitas tarefas na situação “verificar” causava atraso na implantação (*deploy*) de outras funcionalidades, comprometendo assim o ritmo e a freqüência de entregas. Esse

acúmulo era um fato que ocorria devido à atenção que a equipe dava às tarefas urgentes que chegavam a todo instante.

Estava cada vez mais claro que o processo aplicado à época não se mostrava factível à realidade do projeto. Práticas como refatoração, TDD e programação em par eram as primeiras que deixavam de ser aplicadas, para que fosse possível dar uma resposta rápida ao cliente. Mesmo sabendo que deixar essas práticas de lado poderia causar problemas futuros, a equipe não tinha outra opção, pois o cliente esperava uma resposta rápida, de forma que não parasse o nível operacional por conta de *bugs* e ajustes importantes no sistema.

Por conta das características citadas anteriormente foi feito uma avaliação das características do projeto em questão, juntamente com as características dos principais processos ágeis considerados pela empresa. A Tabela 1 mostra um comparativo entre as prescrições do Scrum, Kanban e XP relacionadas com as características de um projeto de manutenção de software.

Tabela 1. Comparativo entre as características de um projeto de manutenção de software com as prescrições do Scrum, Kanban e XP

Características de um projeto de manutenção de software	Scrum	Kanban	XP
Pode haver necessidade de adicionar novos itens de trabalho durante a iteração	Cronograma dividido em iterações *time-box* ou ciclos (*sprints*)	Trabalho orientado a eventos em vez de limite de tempo	Não pode ser implantado sem a aplicação de todas as práticas prescritas como programação em par, TDD, integração contínua e revisão de código
Escopo variável, determinado pelo surgimento de novas demandas	Escopo fixo por iteração	O escopo pode ser alterado constantemente	
As prioridades podem mudar rapidamente	Entrega no final do *sprint* de acordo as prioridades estabelecidas previamente	As prioridades podem ser reajustadas, sem que haja necessidade de se esperar o final de uma iteração	
Dificuldade de estipular itens entregáveis	Itens podem ser quebrados para que eles possam estar completos dentro de um *sprint*	Nenhum tamanho de item em particular é prescrito	

Com o passar do tempo e diante de tais evidências, tornou-se cada vez mais explícito que a natureza do projeto (que estava diretamente ligada à manutenção) não era aderente aos ditames do Scrum ou XP. No início de cada iteração a equipe estava motivada com o planejamento realizado e todos focados em executar as tarefas estimadas. Entretanto, no decorrer do *sprint*, com o aumento das solicitações não planejadas, a equipe tinha que mudar o foco do mesmo para resolver demandas que não haviam sido planejadas. O isolamento da equipe estava se tornando cada vez mais difícil

e todos estavam se desmotivando cada vez mais devido ao não cumprimento dos objetivos acordados nas reuniões de planejamento.

Para definir uma nova abordagem de acompanhamento de projeto, foram identificadas algumas características essenciais para um novo modelo de processo:

- Facilidade de implantação;
- Identificação de gargalos;
- Foco em melhorias pontuais do processo;
- Obtenção de resultados rápidos e mensuráveis.

Com o intuito de desenvolver um fluxo de trabalho bem definido, que se alinhasse com a realidade do projeto, foi decidida a adoção de técnicas do método Kanban para o acompanhamento do projeto, uma vez que ele se mostrou mais adequado às características do projeto, conforme exibido na Tabela 1.

4. O método Kanban

O método Kanban surgiu no Japão com o *Toyota Production System* ou simplesmente TPS [Ohno 1997] para controlar a fabricação de automóveis. Diferente das indústrias americanas, os japoneses optaram por implementar um sistema de produção diferente, onde a demanda sinaliza quando deve-se produzir mais (*pull system*). Com isso, é a demanda que dita o ritmo de produção, fazendo com que a indústria adapte sua velocidade de produção de acordo com o nível de consumo dos clientes.

O uso do método Kanban para o gerenciamento de equipes de desenvolvimento de software registrou um aumento a partir de 2007, quando Rick Garber e David J. Anderson publicaram nas conferências "Lean New Product Development" e "Agile 2007", os resultados obtidos no uso deste método no desenvolvimento de software. Desde então, o uso deste método vem sendo estudado e experimentado por equipes de desenvolvimento.

Kanban é um termo japonês que significa sinal visual. Uma das grandes características deste método é evidenciar os problemas existentes no processo. Das metodologias para desenvolvimento de software, o Kanban é a menos prescritiva, característica essa que estimula ainda mais as equipes a adotarem esse método. Segundo Kniberg (2009) o Kanban tem apenas três prescrições:

- Visualize o fluxo de trabalho atual;
- Limite o fluxo de trabalho;
- Acompanhe e gerencie o fluxo de trabalho.

Sendo um método pouco prescritivo, Kanban acaba tornando-se muito adaptativo. Com isso as equipes que o adotam precisam estar atentas ao processo aplicado para visualizar locais de melhoria e adaptações para que o processo possa fluir de forma satisfatória.

Para atender a primeira prescrição do Kanban é importante ressaltar que o fluxo de trabalho a ser visualizado deve ser aquele que de fato ocorre e não o que formalmente

é definido pela organização. Em muitos locais, apesar de haver um processo oficial, geralmente as pessoas seguem outro modelo.

Já a segunda prescrição, para limitar o fluxo de trabalho (também chamado de WIP - *Work in Progress*) é necessário explicitar quantos itens de trabalho devem estar em cada uma das fases do processo. Esse artifício é um dos pontos-chave do método, uma vez que ele é o responsável por definir Kanban como *pull system*. Apenas quando um item sair de uma fase é que esta fase poderá receber outro item [Kniberg 2009].

Na terceira prescrição, é definida uma forma de medir e controlar o fluxo de trabalho. Neste ponto é onde os times de desenvolvimento realizam adaptações fortes para, de acordo com os problemas evidenciados, propor formas de controlá-los e contorná-los.

Kanban por ser mais adaptativo do que prescritivo, acaba se tornando bastante empírico. As fases do processo em questão e os valores limitados de itens de trabalho para cada fase devem ser testados pela equipe de forma a encontrarem o valor ideal do WIP. Não há uma fórmula para chegar a esse valor, a equipe deve experimentar e encontrar os números que melhor se encaixem na sua realidade.

Dadas as prescrições, percebe-se que o Kanban baseia-se em um processo onde o fluxo de trabalho é contínuo, ou seja, diferente de Scrum, não há um ciclo de trabalho definido onde, conhecida uma atividade, a mesma é estimada e colocada neste ciclo. Kanban controla as entradas de itens de trabalho e a vazão que é dada de acordo com o WIP definido. A esta vazão dos itens de trabalho dá-se o nome de *leadtime*, que representa o tempo de um item de trabalho desde a sua entrada no fluxo de trabalho mapeado até a sua saída.

Kanban permite a combinação de ferramentas de diversos métodos até se obter o processo adequado. Baseia-se fortemente no pensamento Lean [Poppendieck 2003] e estimula a melhoria contínua do processo, de forma a tornar possível dar respostas rápidas ao cliente.

5. Implantação

Com base nas necessidades e problemas encontrados ao longo da execução dos *sprints* e para otimizar o acompanhamento do projeto, foi criado um processo ágil baseado em práticas definidas no framework do Scrum, combinadas com as características do método Kanban.

As tarefas ou itens de trabalho foram representadas por meio de cartões (*post-its*) fixados em um quadro (*cardwall*). Esse quadro, por sua vez, era dividido em colunas que representavam as fases do fluxo de trabalho (*workflow*) da equipe. As tarefas eram distribuídas sequencialmente nas colunas à medida que avançavam no fluxo de trabalho.

No início, foi utilizado um *workflow* simplificado para sinalizar o andamento da iteração, a intenção era começar da forma mais simples possível e aperfeiçoar gradualmente, com consequentes refatorações do quadro. O fluxo era composto por três colunas identificadas pelos seguintes estados:

- **TO DO**: Tarefas elegíveis para entrarem em execução.
- **DOING**: Tarefas em andamento.

- **DONE**: Tarefas concluídas.

Posteriormente, para evoluir essa abordagem, foi feito um levantamento do real escopo do processo que se estava buscando otimizar, para isso, foram identificadas todas as etapas compreendidas entre o momento em que o cliente faz uma solicitação e quando a entrega dessa solicitação realmente acontece. Foi desenhado um esquema contendo todas as etapas desse processo. A construção desse esquema é denominado de mapeamento de cadeia de valor (*Value Stream Mapping*) [Anderson 2010]. Foram identificados os seguintes estados das tarefas:

- **Próximas**: Tarefas elegíveis para entrarem em execução;
- **Em andamento**: Tarefas em andamento;
- **A implantar**: Tarefas concluídas e aptas a serem implantadas no sistema em produção;
- **Implantadas**: Tarefas já implantadas e em uso pelos clientes;
- **Notificadas**: Tarefas concluídas, mas que não precisam ser implantadas, por conta de sua natureza (ajustes no banco de dados, apoio operacional).

Para melhorar a visualização das tarefas no quadro, foi adotada uma classificação estabelecida pela equipe com base nos tipos de solicitações. Essas solicitações foram divididas em categorias ou classes de serviço. Para tornar a divisão das tarefas aparente, cada classe de serviço era representada por um *post-it* com uma cor específica, assim sendo, tornava-se fácil identificar a proporção de tarefas de um determinado tipo que estavam sendo executadas, bastando para isso observar no quadro a predominância da respectiva cor. As cores escolhidas foram:

- **Relatórios *ad hoc*:** verde**;**
- **Correção de *bugs*:** vermelho**;**
- **Novas funcionalidades:** laranja**;**
- **Ajustes operacionais:** amarelo**.**

Com relação ao tempo de execução, as tarefas passaram a ser estimadas com base em um padrão de tamanho denominado *T-Shirt sizing* [Kniberg 2009], onde os tamanhos são atribuídos às tarefas conforme à seguinte regra:

- **Pequeno (P):** Tarefas cuja duração média era inferior ou igual a 1 dia;
- **Médio (M):** Tarefas com duração média superior a um dia e inferior a uma semana;
- **Grande (G):** Tarefas com duração média maior ou igual a uma semana.

De acordo com Kniberg (2009), aumentar o número de atividades em andamento leva diretamente ao aumento do tempo de espera. Por isso, faz-se necessário medir e controlar o número de cartões em uma coluna (WIP) ao longo do tempo. No entanto, em Kanban não é prescrita nenhuma ferramenta para acompanhar o andamento de uma iteração. Para preencher essa lacuna, lançou-se mão do Diagrama de Fluxo Cumulativo

(DFC), pois, segundo Kniberg (2009) um DFC ajuda a fornecer os incentivos corretos para minimizar absolutamente o número de itens esperando em filas.

O DFC (Figura 1) é uma ferramenta visual composta por dois eixos, um horizontal (eixo X) e um vertical (eixo Y), onde no eixo Y é colocado diariamente o total do número de itens em cada coluna do quadro. O eixo X representa os dias da iteração. Seu objetivo é mostrar uma imagem de todo o processo e comunicar a capacidade produtiva da equipe.

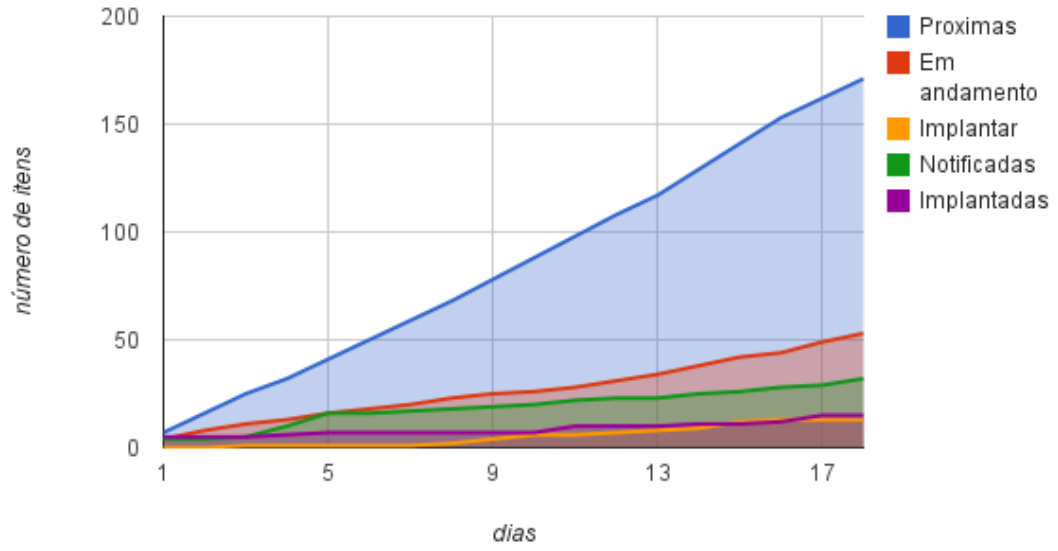

Figura 1. Diagrama de fluxo cumulativo.

Algumas boas práticas ágeis foram mantidas, outras foram adaptadas. As reuniões diárias, por exemplo, continuaram a ser realizadas no início de cada dia com a equipe posicionada em frente ao quadro de tarefas, proporcionando uma melhor contextualização das atividades que estavam sendo realizadas por cada membro da equipe. As reuniões de planejamento passaram a ter um caráter mais de priorização do que de estimativa de tarefas. Retrospectivas e revisões ainda são realizadas em um intervalo de duas semanas.

A aplicação da abordagem foi executada pelos autores, os quais também foram responsáveis pela apresentação, para todos os colaboradores do setor de tecnologia da empresa, dos conceitos e ferramentas que foram utilizados.

6. Dificuldades e benefícios encontrados

6.1. Dificuldades

Inicialmente foi observada dificuldade em perceber os benefícios da utilização do Kanban, principalmente por este não interferir diretamente no trabalho da equipe.

Dificuldades na monitoração do projeto também foram observadas devido ao fato das pessoas responsáveis pelo acompanhamento do processo estarem também envolvidas nas tarefas em andamento, causando um sobrecarga de atribuições em alguns momentos, principalmente durante a execução de tarefas críticas.

Outra dificuldade foi o fato da empresa possuir sua própria ferramenta de acompanhamento de projetos e a mesma ter sido concebida fortemente baseada nos conceitos do Scrum, dificultando a gestão de projetos sob a ótica de outra metodologia.

6.2. Benefícios

A Tabela 2 mostra um comparativo entre a primeira iteração Kanban e a sexta iteração, quando algumas mudanças já haviam sido aplicadas, motivadas pelos sinais apresentados com o uso do Kanban.

Tabela 2. Comparativo da primeira e da sexta iteração utilizando o Kanban.

	1ª Iteração	6ª Iteração
Quantidade de desenvolvedores na equipe	4	5
Quantidade média de tarefas entregues (por desenvolvedor)	6	8
Quantidade média de tarefas entregues (equipe)	24	40
WIP (*Work in Progress*)	10	10
Leadtime (dias)	15	6

Os números mostram que houve uma melhora na quantidade média de tarefas entregues por desenvolvedor. Esse fato explica-se por dois motivos:

- Durante as iterações, foi percebido um dos grandes problemas: após a finalização de uma atividade, acumulavam-se várias tarefas para serem disponibilizadas no sistema. Isso ocorria devido à equipe, após finalizar a tarefa, partir para a resolução de outra, ficando a atividade de liberação em segundo plano. Ao acumular muitas tarefas, a atualização do sistema se tornava mais onerosa, demandando mais tempo para atividade. A decisão foi retirar da responsabilidade dos desenvolvedores tal atividade. Esta tarefa ficaria restrita ao coordenador do projeto e à equipe de suporte e infra-estrutura.
- No início, o quadro de tarefas estava ficando cada vez mais vermelho (a cor que se convencionou para tarefas de correção de *bugs*). Ao analisar o tipo dessas tarefas, verificou-se que muitos *bugs* poderiam ser evitados se houvesse uma melhor discussão sobre como a tarefa deveria ser implementada, evitando assim retrabalho. Ao exigir maior presença do cliente para discussão das atividades e maior integração do coordenador do projeto para mediar tais discussões, foi possível perceber um aumento na quantidade de tarefas entregues por desenvolvedor durante uma iteração. Além disso, a verificação de tarefas pelo coordenador passou a ser mais rígida, evitando assim que fossem liberadas funcionalidades incompletas que futuramente voltariam como *bugs*.

Além disso, na Tabela 2 pode-se notar que o WIP não se alterou. Apesar do fato da equipe ganhar mais um desenvolvedor, optou-se por manter o WIP com o valor

inicial justamente para possibilitar uma redução no *leadtime*, uma vez que o tempo de resposta lento era considerado um dos maiores problema da equipe na percepção dos seus clientes. Com essa estratégia, o *leadtime* caiu de 15 (quinze) para 6 (seis), ou seja, a previsão média para cada nova tarefa que chegava eram de 6 dias para ser concluída.

Os resultados quantitativos do uso do método ratificam a análise qualitativa apresentada a seguir. Dentre os resultados qualitativos considerados positivos obtidos durante a implantação da abordagem podem-se citar:

- O mapeamento do fluxo de trabalho condizente com a realidade, tendo em vista que o quadro de tarefas passou a refletir não as pessoas fazendo o trabalho, mas sim o fluxo real de trabalho que está sendo feito.
- A visão crítica do fluxo de trabalho existente, focando na melhoria contínua. Dessa forma foi possível criar um processo estável e constantemente otimizado, que prioriza a visibilidade do trabalho em execução, evidenciando impedimentos e identificando gargalos.

Com essa experiência também foi possível uma disseminação do conhecimento sobre gerência de projetos em toda a organização, obtida pelo compartilhamento dos erros e acertos vividos pela equipe do projeto.

Constatou-se que, a fácil visualização do processo e as ferramentas de acompanhamento e medição forneceram à equipe um maior poder de argumentação frente aos *stakeholders*. Uma evidência disso foi que a partir do acompanhamento visual da execução do processo e do diagrama de fluxo cumulativo (Figura 1) foi possível convencer a gerência de tecnologia de que a equipe realmente precisava de mais desenvolvedores, pois o gráfico atestava a discrepância entre a capacidade de trabalho do time (a linha roxa do gráfico) e a crescente demanda dos clientes (linha azul).

Uma grande vantagem dessa abordagem foi que as demandas urgentes deixaram de ser uma ameaça ao bom andamento do fluxo de trabalho. Ao invés disso, elas passaram a fazer parte dele, pois a equipe encontra-se sempre preparada para o aparecimento dessas solicitações críticas de última hora, uma vez que, quando algo inesperado surge, simplesmente põe-se uma tarefa no quadro com alta prioridade e a mesma é executada pelo primeiro desenvolvedor que estiver disponível, de tal forma que o fluxo vai correndo naturalmente, sem que haja necessidade de interrupção do mesmo.

7. Limitações

A experiência aqui relatada baseou-se exclusivamente na realidade de um projeto de uma única empresa. A falta de aplicação da abordagem em outras equipes de empresas diferentes, com processos de desenvolvimento distintos é uma limitação clara do trabalho. Uma maior diversidade de exemplos da aplicação dessa abordagem poderia auxiliar os interessados em usá-la.

8. Trabalhos Relacionados

Ladas (2008) apresenta como integrar Scrum e Kanban para o gerenciamento de *softwares* obtendo agilidade. Além deste, destaca-se o trabalho de Kniberg (2009), no qual é mostrado como combinar os dois métodos, evidenciando diferenças e

semelhanças. Entretanto, pouco se fala em relação à manutenção de softwares em produção com a utilização de Kanban.

9. Conclusão

Este trabalho relata experiências relacionadas à adoção de processos para controlar a manutenção de software. Foi utilizado o Kanban como processo para organizar o trabalho de uma equipe de desenvolvimento que apresentava dificuldades em usar o Scrum para controlar suas atividades.

Com a aplicação do Kanban foi possível obter resultados significativos relativos à melhoria de processo, tornando-o mais adequado às características do projeto. Observou-se pouca resistência ao processo de implantação, causando um impacto mínimo na forma de trabalhar de cada desenvolvedor. A frustração causada pelo insucesso dos *sprints* no Scrum deu lugar a um aumento na motivação dos membros da equipe com os resultados alcançados e a sensação de entrega contínua.

O uso do Kanban para projetos de manutenção na empresa Infoway obteve os melhores resultados, quando comparados ao Scrum e XP. Suas características se mostraram apropriadas para projetos de manutenção de software e seu uso deve ser considerado por outras empresas que atuem nessa área e que já tenham se deparado com alguns dos problemas aqui relatados.

Referências

Anderson, David J. *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, 2010.

Beck, Kent. Programação Extrema Explicada: Acolha as mudanças. Porto. Alegre: Bookman, 2004.

IEEE. Std 1219 – *IEEE Standard for Software Maintenance***,** Institute of Electrical and Electronics Engineers, 1998.

ISO/IEC 14764. *Software Engineering - Software Maintenance*, ISO and IEC, 1999.

Kniberg, H., *Kanban vs. Scrum: Making the most of both*, 2009, http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf [18-Jan-2010].

Ladas, C., *Scrumban*. Seattle, WA, USA: Modus Cooperandi Press, 2008.

MCT. *Programa Brasileiro da Qualidade e Produtividade de Software*. Ministério de Ciência e Tecnologia, 5ª edição, 2009.

Ohno, T. *O Sistema Toyota de Produção*. Porto Alegre: Bookman, 1997.

Pfleeger, S. L. *Software Engineering – Theory and Practice*, 2ª edição. Prentice-Hall, 2001.

Pigoski, T. M. *Pratical Software Maintenance: Best Practices for Managing your Software Investment*. Wiley Computer Publishing, 1996.

Poppendieck, M., Poppendieck, T. Lean Software Development: An Agile Toolkit. Addison Wesley, 2003.

Schwaber, K., Beedle, M. *Agile software development with Scrum*. Prentice Hall, 2002.