ISSN: 2175-1897

Análise das características do kanban e de possíveis procedimentos aplicáveis a métodos de produção de software

Raquel Bortoluci
Centro Paula Souza – SP – Brasil
raquelborto@gmail.com

Valter Yogui
Centro Paula Souza – SP – Brasil
valter.yogui@fatecsp.br

Marilia Macorin de Azevedo
Centro Paula Souza – SP – Brasil
marilia.azevedo@fatec.sp.gov.br

Marcelo Duduchi
Centro Paula Souza – SP – Brasil
mduduchi@gmail.com

Resumo - O objetivo deste trabalho é analisar as características do Kanban, suas propriedades e identificar possíveis procedimentos a serem aplicados com os métodos de produção de software Cascata, Scrum e XP. Com este trabalho, observa-se que para aplicar o Kanban é necessário que se tenha um processo estabelecido, baseado em propriedades básicas, como visualização do fluxo de trabalho, trabalho em progresso limitado, fluxo medido e gerenciado, políticas explícitas de processos e uso de modelos para reconhecer oportunidades de melhoria. Observa-se que o Kanban pode ser empregado em processos já utilizados pelas empresas no desenvolvimento de software quando utilizando métodos de desenvolvimento de software como o Scrum e o XP.

Palavras-chave: Kanban, *just-in-time*, engenharia de software, cascata, Scrum, XP

Abstract - The objective of this study is to analyze the Kanban features and their properties and identify possible procedures to be applied to the methods of software production Waterfall, Scrum and XP. With the research, it was possible to verify that to apply Kanban is necessary an established process based on basic properties as visualization workflow, working in limited progress, managed and measured flow; explicit policies of processes and use models to recognize improvement opportunities. It was also verified that the Kanban can be used with existing process already followed by the companies when using development method such as Scrum and XP.

Keywords: Kanban, just-in-time, software engineering, waterfall, Scrum, XP

1. Introdução

O Kanban, que em japonês significa literalmente “registro ou placa visível”, introduzido junto com o Sistema Toyota de Produção, foi desenvolvido por Taiichi Ohno, então Vice-Presidente da Toyota Motor Company, tornando-se um método enraizado na empresa para que perpetuasse uma das principais características deste sistema: a produção *just-in-time* (JIT), fator primordial numa indústria de montagem. Segundo o próprio Ohno (1997), o sistema Toyota de produção desenvolveu-se a partir de uma necessidade: as restrições de mercado tornaram necessária a produção de pequenas quantidades de muitas variedades de produtos, e sob condições de baixa demanda. Sobre a filosofia JIT, uma característica fundamental, conforme Sugimori *et al.* (2007), é que este é um sistema de respeito com o aspecto humano, onde os trabalhadores podem exibir suas capacidades a partir da participação ativa na execução e melhoria das suas próprias atividades. Monden (2012) define JIT como um processo que produz peças necessárias em quantidades necessárias no tempo necessário.

Pelo lado da Engenharia de Software, para que esta gere valor e contribua para o sucesso das empresas por meio do desenvolvimento de sistemas de informação, temos métodos que podem ser classificados como convencionais ou ágeis, mas que têm em comum o mesmo conjunto de premissas nas suas propostas principais. Conforme Pressman (2006) e Sommerville (2007), é necessário: ter o escopo bem definido, ou seja, deve-se estabelecer o problema a ser resolvido e as necessidades a serem atendidas, dentro das condições determinantes para um atendimento satisfatório; a especificação do sistema, que corresponde a descrição das funções do software, de modo a implementar a solução do problema, o cumprimento e o atendimento das necessidades, e que deve ser uma forma detalhada para permitir o desenho do sistema, sua construção e respectivo teste, da forma mais fidedigna possível; e a relação existente entre as duas definições anteriores, onde a complexidade do escopo determina os prazos e os custos do desenho do sistema, sua construção e seus testes.

Anderson (2010) observa que para utilizar o Kanban é necessário existir um processo estabelecido. Observa-se que a popularidade do método vem crescendo (ANDERSON; ROOK, 2011) devido aos benefícios que este pode trazer ao desenvolvimento de software (IKONEN *et al.*, 2011), mas que também ainda existem divergências quanto a classificação do Kanban no contexto de desenvolvimento de software (MAEDA, 2011). Os fatores acima citados mostram que o Kanban tem se apresentado bastante promissor como um método no processo de desenvolvimento de software e por isso este artigo busca justificativas para a sua utilização dentro de métodos de produção de software já estabelecidos no mercado.

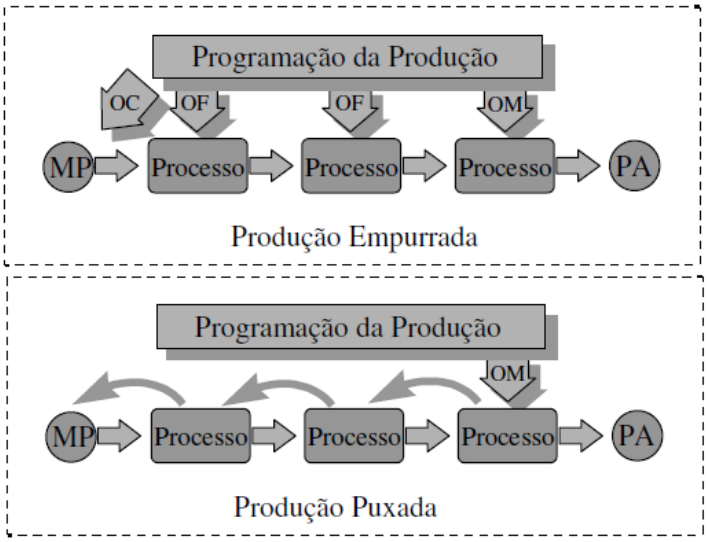
2. Referencial Teórico

A produção sob a filosofia JIT é formada por duas premissas básicas: a melhoria contínua, pressupondo o JIT como um processo contínuo, e a eliminação do desperdício, minimizando todas as atividades que não agregam valor direto ao produto ou serviço. Para alcançar estas premissas é necessário utilizar uma ferramenta que permita gerenciar o fornecimento de materiais nos postos de

trabalho, no momento certo. Esta ferramenta pode ser o sistema Kanban para controle da produção, que é baseado no sistema de puxar e usado como instrumento de controle da produção a partir do gerenciamento do estoque de produtos em processo, abastecendo todos os processos com informações em tempo real. O sistema de puxar trata o processo de produção na perspectiva do produto acabado, considerando que suas ordens representam requisições concretas dos clientes.

A produção puxada, ao contrário da produção convencional, empurrada ou prevista, tem por objetivo utilizar os recursos produtivos disponíveis da maneira mais racional possível, de tal forma que o fluxo produtivo seja maximizado, ao invés do foco nas capacidades individuais. No caso citado da Toyota, um sistema de cartões de produção, o Kanban, foi utilizado para solicitar as peças necessárias, onde os operadores têm somente autorização para produzir peças quando recebem o cartão Kanban. A figura 1 apresenta as diferenças entre a produção puxada e a produção empurrada:

Figura 1: Produção empurrada versus Produção enxuta



Fonte: extraído de SALOMON - DPD/FEG/ UNESP 1999-2002

Como toda ferramenta, o Kanban pode apresentar algumas limitações e desvantagens. É um sistema destinado à produção repetitiva e, portanto, é necessário ter uma programação nivelada, contêineres (postos de armazenagem) padrões, grande cooperação dos fornecedores e uma disciplina muito rígida. Enfatiza a tecnologia de processo, como nos produtos baseados na configuração do fluxo de produção e, quando bem implementado, traz vantagens como o aumento da produtividade, em conjunto com a redução de estoques e dos tempos intermediários de produção. Em função do projeto do produto e do fluxo do sistema de produção, permite à empresa responder às pequenas e previsíveis variações do mercado. O Kanban é um sistema simples de controle de fluxo com ênfase no controle de estoque do modo mais simples, envolvendo reduzido trabalho administrativo comparado com os outros sistemas, de modo a permitir a determinação de prioridades.

ISSN: 2175-1897

A filosofia JIT pode ser exercida sem o uso de ordens de produção para controlar os componentes em cada unidade de trabalho. Neste caso, o Kanban servirá como uma ordem de produção, administrando o processo para que haja a produção dos componentes na hora certa. Na prática, o Kanban é geralmente determinado por um cartão de papel, existindo também outros tipos de sinais que podem ser utilizados, tais como sinais eletrônicos ou luzes. Existem dois tipos principais de cartões: Kanban de Produção: cartão que define a quantidade de um componente específico que uma unidade de trabalho (que produz o item) deveria produzir para repor o que foi retirado e utilizado; e Kanban de Movimentação: cartão que autoriza a movimentação de material pela fábrica, circulando entre a unidade de trabalho de produção e o seu posto de armazenagem (contêiner) junto à unidade de trabalho consumidora. Cada contêiner contém um cartão de produção ou de movimentação.

O Kanban, apesar de ser considerado um método dentro da filosofia JIT, não pode ser catalogado imediatamente como um possível método de desenvolvimento de software ou uma abordagem de gerenciamento de software. Para se aplicar o Kanban é necessário que se tenha um processo estabelecido (ANDERSON, 2010). O Kanban é baseado em cinco propriedades básicas: visualização do fluxo de trabalho, trabalho em progresso limitado, fluxo medido e gerenciado, políticas de processos explícitos e uso de modelos para reconhecer oportunidades de melhoria (ANDERSON, 2010). No contexto de software, Anderson e Rook (2011) destacam, ainda, que Kanban é um método para o avanço incremental, de mudanças evolucionárias que limitam o trabalho em andamento e criam um sistema de tração, no qual um novo trabalho só pode ser iniciado quando o trabalho anterior esteja concluído.

Maeda (2011) observa que Anderson publicou em 2008 a primeira abordagem do Kanban no desenvolvimento de software. O autor afirma que o Kanban vem sendo chamado por muitos como método de desenvolvimento ágil (segunda geração ágil), ou como um método *lean*, ou *leanagile*. Nota-se, portanto, que ainda existem divergências sobre a classificação do Kanban no desenvolvimento de software.

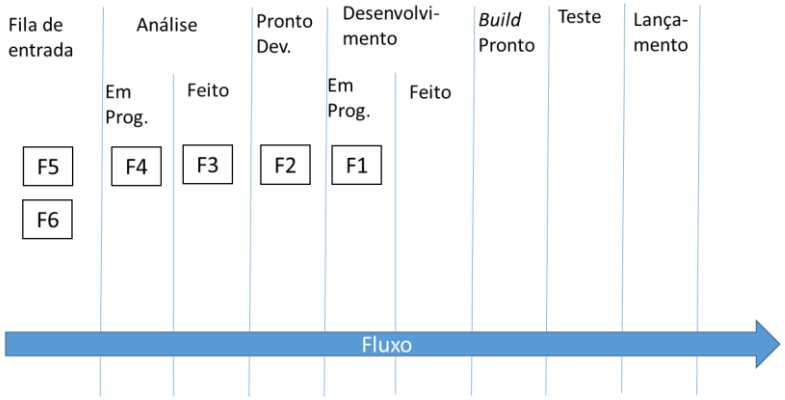
Para se implementar o Kanban no desenvolvimento de software deve-se mapear o fluxo de trabalho existente, definindo-se um ponto de início e um ponto de fim para controles, que serão considerados pontos de interface entre o começo e o fim. Por exemplo, quando se tem o controle sobre a análise, o projeto, a codificação e o teste, deve-se então mapear estes pontos de valores e negociar novas formas de interação com os parceiros de negócios, a fim de adaptar as formas de adquirir os requisitos, a priorização de funcionalidades e o gerenciamento de portfólio (ANDERSON, 2010).

Ikonen *et al.* (2011), em um estudo empírico, observam que a utilização do Kanban pode trazer vários benefícios no desenvolvimento de software. Os autores notaram que a utilização do método proporcionou a redução de documentação, no qual a equipe apenas escreveu documentos necessários, sem perder tempo no preparo de documentos desnecessários. Relatou-se, também, que houve melhoria na resolução de problemas, pois estes eram resolvidos assim que surgiam. Quanto à visualização do fluxo de trabalho, este formato ajudou a manter o time motivado e auxiliou o mesmo a selecionar as funcionalidades do produto para demonstração,

além de manter todos os envolvidos cientes quanto ao progresso e os problemas encontrados durante o processo de desenvolvimento. Com a utilização do Kanban foi possível que todos tivessem o entendimento do todo e pudessem atribuir tarefas de maneira mais efetiva, além de ajudar na comunicação e no *feedback* do trabalho, pois a visualização tornou mais claro o andamento do projeto. Notou-se, também, que o método foi intuitivo para a equipe e por isso de fácil aceitação.

Para se implementar o Kanban, deve-se ter a visualização do fluxo de trabalho. Esta visualização de trabalho traz mais transparência à equipe que pode ter visibilidade de qualquer item de trabalho, observar em que cada membro está trabalhando, permitindo que melhor se apropriem os recursos, identifique gargalos e ajude os gestores a tomar a decisões (ANDERSON, 2010). A visualização de trabalho é feita com um painel Kanban, onde se registra o avanço do projeto, cada coluna indica o status das tarefas e os cartões devem ser posicionados na coluna de acordo com o status, conforme a figura 2:

Figura 2: Fluxo de trabalho - Kanban



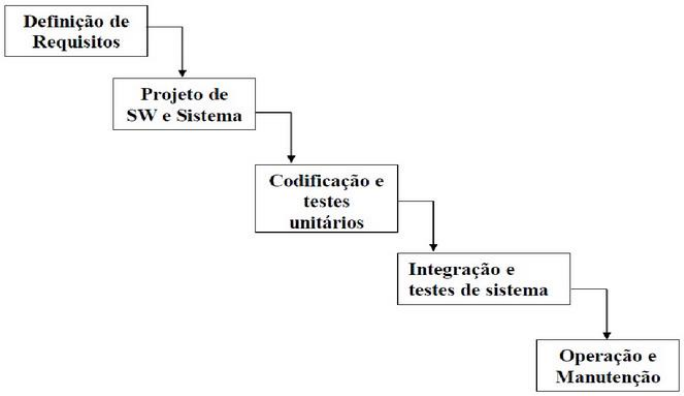
Fonte: adaptado de ANDERSON (2010)

A fim de identificar onde o Kanban poderia ser usado no desenvolvimento de software, serão descritos os fluxos de trabalho dos métodos Cascata, como exemplo de um método tradicional de produção de software, o Scrum e o XP, como exemplos de métodos ágeis para desenvolvimento de software.

Para Pressman (2006), o método Cascata é um processo que segue linearmente as etapas de engenharia do sistema: análise de requisitos, projeto, geração de código, testes e manutenção. Em cada uma destas etapas, um conjunto pré-estabelecido de atividades é realizado de forma que os artefatos produzidos em cada etapa sirvam de entrada para a etapa seguinte. Para Sommerville (2007), o fluxo de trabalho no desenvolvimento de software em Cascata segue uma sequência de etapas, na qual uma etapa é seguida da outra. As principais etapas do modelo em Cascata são: análise e definição de requisitos, projeto de sistema e software, implementação e teste de unidade, integração de teste de sistema, e operação e manutenção. Na análise e definição de requisitos define-se os requisitos por meio de consulta aos usuários do sistema e cria-se uma documentação. Na fase de projeto de sistema e software estabelece-se uma arquitetura geral do sistema. Na implementação e teste de unidade realiza-se o

projeto de software e executa-se o teste unitário. Já na fase de integração de teste de sistema, integra-se as unidades individuais ou programas e testa-se o sistema como um todo. Na operação e manutenção instala-se e coloca-se o sistema em operação. A manutenção envolve a correção de erros não detectados nos estágios de teste e em melhorias no sistema.

Figura 3: Esquema gráfico do Modelo Cascata



Fonte: adaptado de SOMMERVILLE (2007)

No Scrum o fluxo de trabalho funciona da seguinte forma: todos os requisitos são listados e priorizados em uma lista chamada *product backlog*. Posteriormente, uma lista chamada de *sprint backlo*g é definida para que estes itens sejam desenvolvidos dentro de um *sprint,* isto é, em pequenas iterações compostas por atividades de design, desenvolvimento, teste, documentação e outras atividades pertinentes ao desenvolvimento e finalização completa de um produto. Durante toda a execução do projeto, uma reunião diária de cerca de quinze minutos é organizada a fim de verificar o progresso e identificar itens que requerem mais atenção ou até mesmo ajuste no *sprint backlog*. No final de cada sprint é feita uma reunião para revisar o resultado alcançado e uma reunião de retrospectiva para o time analisar o trabalho desenvolvido (HANSMANN; STOBER, 2010).

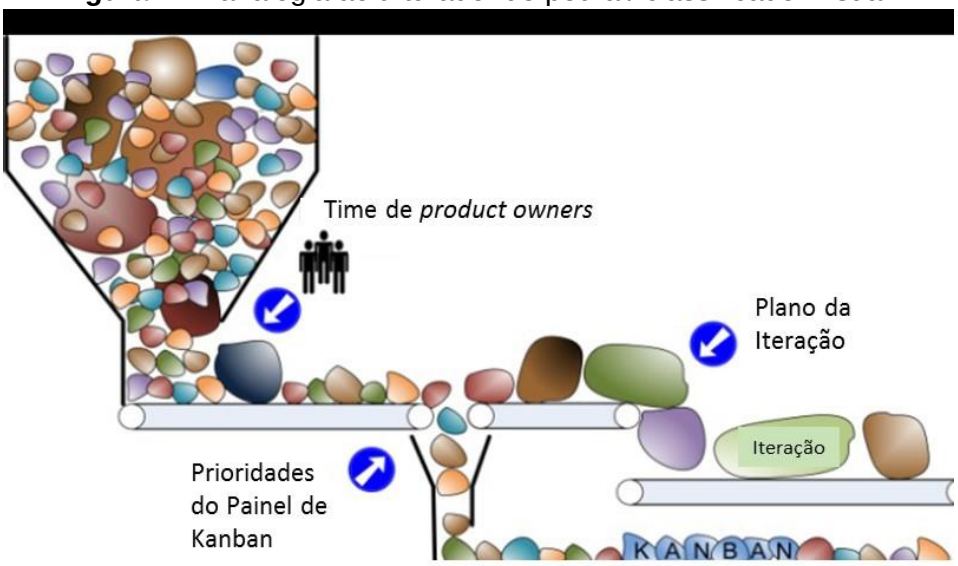
Segundo Schwaber e Sutherland (2013), no Scrum a equipe de desenvolvimento trabalha unida com o objetivo de entregar o software e tem autonomia para definir a tática para esta consecução. O Scrum atua principalmente na gerência do projeto, sem determinar como a equipe executará as tarefas de programação. Esta abordagem favorece a auto-organização da equipe e permite a integração com outros métodos ágeis que foquem nas práticas de programação, como por exemplo, o próprio XP. Três papéis estão presentes no método: o *Product owner* - deve possuir a visão do produto em vários níveis; a Equipe - é uma unidade dentro do método e deve ser multifuncional e autossuficiente; e o *Scrum Master* - deve possuir conhecimento de todo o processo para garantir que ele seja seguido e deve manter uma ampla visão sobre o projeto. O *Scrum* caracteriza-se como um processo empírico e adaptativo, com as fases: planejamento, *sprint* e avaliação.

No XP, Pressman (2011) destaca que o desenvolvimento de software deve ser composto por quatro atividades: planejamento, projeto, codificação e testes. O planejamento inicia-se com o processo de ouvir, o qual tem por objetivo levantar e entender os requisitos dos clientes, resultando na criação de histórias. Para o desenvolvimento do projeto segue-se o princípio da simplicidade no qual se deve

apenas desenvolver o que será necessário para o software. Na atividade de codificação, o desenvolvedor constrói o teste unitário e o código do software. Os testes unitários devem ser desenvolvidos no início da codificação, auxiliando a implantação de testes de regressão, e posteriormente são executados os testes de aceitação ou teste de clientes.

Observa-se que o Kanban pode ser introduzido na rotina dos métodos ágeis. O processo de distribuição de trabalho utilizando Kanban em método ágil (Scrum e XP) pode ser explicado conforme a figura 4, no qual se faz uma analogia à um triturador de pedra (POLK, 2011).

Figura 4: A analogia ao triturador de pedra / classificador visual.



Fonte: adaptado de POLK (2011)

Observa-se que o time de *Product owners* lista, quantifica e prioriza o trabalho em várias partes. Na reunião de planejamento de iteração o time escolhe os itens com maior prioridade. O Kanban, mecanismo de classificação para todos os itens, define tamanho e complexidade (POLK, 2011).

3. Método

Este trabalho é uma pesquisa bibliográfica com objetivo de analisar as características do Kanban e suas propriedades e identificar possíveis procedimentos a serem aplicados com os métodos de produção de software cascata (*waterfall*), *scrum* e XP (*eXtreme Programming*), encontrando as oportunidades para contemplar os princípios do Kanban nos métodos de desenvolvimento de software, ou seja, visualização do fluxo de trabalho, trabalho em progresso limitado, fluxo medido e gerenciado, políticas de processos explicitas e uso de modelos para reconhecer oportunidades de melhoria.

4. Resultados e Discussão

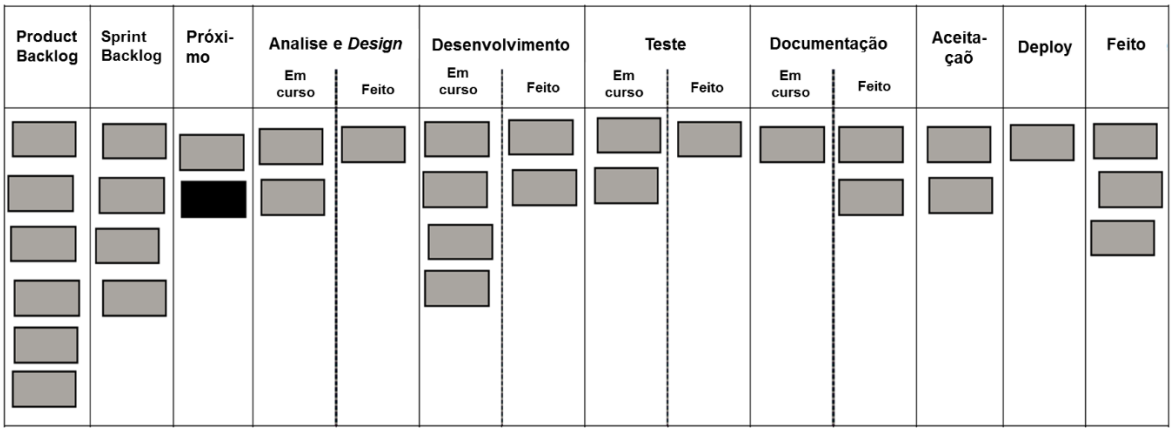
O método cascata, com uma abordagem pragmática, produz o software final a partir da execução de etapas sistematicamente definidas em um processo que as

ISSN: 2175-1897

segue linearmente. Em cada uma destas etapas, um conjunto pré-estabelecido de atividades é realizado de forma que os artefatos produzidos em cada etapa sirvam de entrada para a etapa seguinte, numa demonstração de razoável semelhança à produção empurrada, o que pode dificultar ou tornar sem efeito a utilização do Kanban. Segundo Sommerville (2007), o fluxo de trabalho no desenvolvimento de software em cascata segue uma sequência de etapas, na qual uma etapa é seguida da outra.

O Scrum é um método que já vem sendo utilizado com o Kanban. O Scrumban é a concordância entre o Scrum e o Kanban (Scrumban, 2015). Para demonstrar o trabalho em progresso utilizando o painel de Kanban no Scrum, Mahnic (2014) propõe a organização de colunas como na figura 5.

Figura 5: Painel Kanban no Scrum

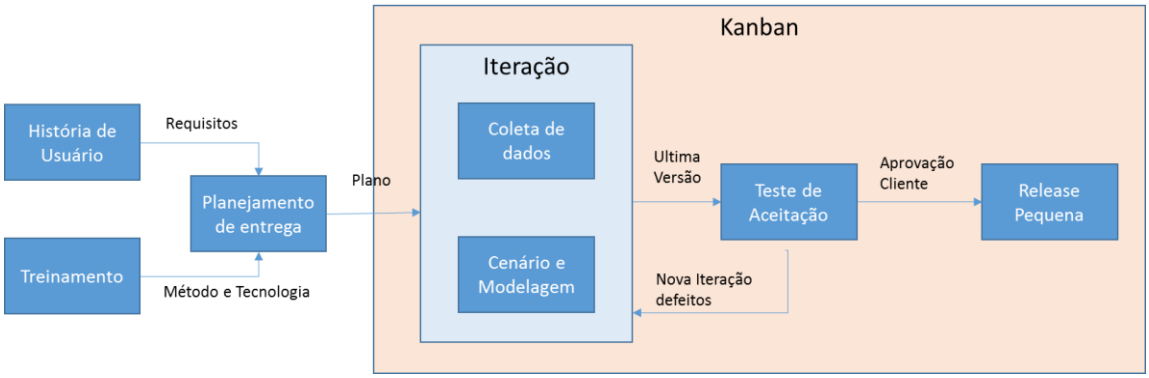


Fonte: adaptado de MAHNIC (2014)

No painel proposto por Mahnic (2014), observa-se que ele é composto por colunas que demonstram o ciclo de desenvolvimento do Scrum, sendo que é composto inicialmente pelo *product backlog*, onde todas as histórias são listadas. Posteriormente, seleciona-se as histórias a serem desenvolvidas dentro de um *sprint*. A coluna *next* tem como objetivo priorizar as histórias mais importantes, e quando o trabalho de uma história termina, o desenvolvedor deve trabalhar nas que estão nesta coluna, por serem mais prioritárias. Após a seleção das histórias, o trabalho é feito pelas atividades de análise e design, desenvolvimento, teste, documentação (arquivos de ajuda ou documentação para o usuário), aceitação (teste final feito pelo *product owner), deplo*y e *done/live* com as histórias entregues aos clientes.

Encontra-se, também, a possibilidade de utilizar o XP com o Kanban; para isso Han e Xie (2012) propõem o modelo da figura 6:

Figura 6: Modificação de processo do método (Extremeban)



Fonte: adaptado de Han e Xie (2012)

Como observado na figura 6, o progresso de trabalho é controlado pelo Kanban depois do planejamento. Os autores sugerem que primeiramente deve-se treinar a equipe para que todos tenham conhecimento sobre o método utilizado, a definição das histórias de usuários e o planejamento, que irá priorizar as histórias a serem desenvolvidas. Na iteração, quando o software é desenvolvido de acordo com os valores do XP, começa a utilização do uso de Kanban para direcionar o fluxo de trabalho, e posteriormente o software será analisado pelos *stakeholders deste processo* e então entregue para uso.

5. Considerações finais

Neste trabalho foi possível verificar que o Kanban pode ser utilizado com alguns métodos de desenvolvimento de software. Os métodos de desenvolvimento ágil, tais como o Scrum e XP, podem ser considerados mais adequados, visto que se aproximam mais do modelo de produção puxada, analogamente à mudança imposta pelo JIT, que utiliza do Kanban como instrumento de controle na indústria de montagem. Com o Kanban foi possível criar uma sequência de trabalho limitada respeitando os valores dos métodos ágeis. Em contrapartida, o método Cascata, a princípio, não se apresenta compatível para a utilização de Kanban, com sua sequência rígida de etapas, sem retorno ou *feedback* de resultados entre as mesmas, assemelhando-se a uma produção empurrada.

Este artigo foi baseado em pesquisas bibliográficas referentes às características do Kanban específicas da indústria de montagem, em métodos de desenvolvimento de software e suas características principais e em artigos que combinam o tema Kanban e a produção de software. Constatam-se oportunidades de pesquisas da utilização do Kanban com outros métodos e em um universo maior de referências. Estudos de casos e pesquisas práticas poderiam ser aplicadas para confirmar se as utilizações dos métodos combinados trariam mais eficiência às empresas.

Referências

ANDERSON, D. J. *Kanban: Successful Evolutionary Change for Your Technology Business*. Washington: Blue Hole Press, 2010.

ANDERSON, D. J; ROOCK, A. An Agile Evolution: Why Kanban Is Catching On in Germany and Around the World. *Cutter IT Journal: The Journal of Information Technology Management*, 2011.

HANSMANN, U; STOBER, T. *Agile Software Development: Best Practices for Large Software Development Projects*. 1. ed. New York: Springer, 2010.

HAN, B; XIE, J. Practical Experience: Adopt Agile Methodology Combined With Kanban For Virtual Reality Disponível em: < https://gupea.ub.gu.se/handle/2077/30036>. Acesso em: 05/07/2015

IKONEN, *et al*. On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation. *16th IEEE International Conference on Engineering of Complex Computer Systems*, 2011.

MAEDA, M, K. Opening Statement. *Cutter IT Journal: The Journal of Information Technology Management*, 2011.

MAHNIC, V. Improving Software Development through Combination of Scrum and Kanban. *Recent Advances in Computer Engineering, Communications and Information Technology*, 2014.

MONDEN, Y. *Sistema Toyota de Produção - uma abordagem integrada ao just-in-time.* Bookman. 2012.

OHNO, Taiichi. *O Sistema Toyota de Produção – Além da Produção em larga escala.* Bookman. 1997.

POLK, R. Agile & Kanban In Coordination. *2011 Agile Conference*, 2011.

PRESSMAN, R. S. *Engenharia de software.* 720p., 6. ed., ISBN 8586804576, ed. McGraw-Hill. 2006.

PRESSMAN, R.S. *Engenharia de Software: Uma abordagem Profissional*. 7. ed. Porto Alegre: AMGH, 2011.

SALOMON, V. A. P.; (1999-2002) - DPD/FEG/UNESP. Disponível em <http://www.feg.unesp.br/~salomon/grad/pcp/pcp3>. Acesso em 07/07/2015.

SCHWABER, K. and SUTHERLAND, J. 2013. *The Scrum Guide™ The Definitive Guide to Scrum - The Rules of the Game.* Disponivel em <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf> - Acesso em 09/06/2014.

Scrumban. A Guide To Scrumban**.** Disponível em: < http://www.aboutscrumban.com/>. Acesso em: 05/07/2015.

SOMMERVILLE, I. *Engenharia de software*. 552 p., 8.ed., ISBN 8588639289, ed. Addison Wesley, 2007.

SUGIMORI, Y.; KUSUNOKI, K.; CHO, F.; UCHIKAWA, S. *Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system.* International Journal of Production Research. pgs 553-564, publicação. 2007.