PUC-MG – PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
DIRETORIA DE ENSINO A DISTÂNCIA

Thales Souza Oliveira

DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA COMPARAÇÃO ENTRE O METÓDO SCRUM E O SISTEMA KANBAN BASEADO NO FRAMEWORK SEMAT ESSENCE.

Brasília
2015

Thales Souza Oliveira

DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA COMPARAÇÃO ENTRE O METÓDO SCRUM E O SISTEMA KANBAN BASEADO NO FRAMEWORK SEMAT ESSENCE.

Trabalho de Conclusão de Curso de Especialização em Engenharia de Software como requisito parcial à obtenção do título de especialista.

Professor (a): Maria Augusta Vieira Nelson

Brasília

2015

SUMÁRIO

1. CONTEXTUALIZAÇÃO DO PROJETO DE PESQUISA

Para o processo de desenvolvimento de software a engenharia de software disponibiliza algumas ferramentas e metodologias para permitir a organização e estruturação no desenvolvimento de software. Podemos citar alguns métodos tradicionais:

- Modelo cascata;
- Modelo de processo incremental;
- Modelo de processo evolucionário;
- Modelo Espiral;
- Modelos concorrentes;
- Processo Unificado (UP).

Esses modelos são apresentados em diversas bibliografias e estudados por diversos autores, sendo que Pressman apresenta a conceituação dos métodos prescritivos:

> Os modelos conhecidos como métodos tradicionais ou prescritivos recebem a denominação de prescritivos porque prescrevem um conjunto de elementos de processo, atividades metodológicas, ações de engenharia de software, tarefas, produtos de trabalho, garantia da qualidade e mecanismos de controle de mudanças para cada projeto. (PRESSMAN, 2011).

No cenário atual dos processos de desenvolvimento de software são enfrentados diversos desafios, entre estes, podemos destacar as mudanças das necessidades dos usuários finais que podem ocorrer frequentemente por um ambiente altamente competitivo entre as organizações, exigência de softwares com uma maior qualidade e limitações de recursos. Além dos aspectos apresentados, os processos tradicionais possuem pontos fracos e desvantagens se aplicados à determinada realidade de algumas organizações e projetos dependendo do contexto de sua aplicação.

Para esse cenário, pode ser proposto a utilização dos métodos ágeis de desenvolvimento, métodos ágeis são processos que suportam a filosofia ágil e que consistem de elementos individuais chamados de práticas. Entretanto o desconhecimento do que realmente é agilidade e de como deve ser aplicado um determinado método de acordo com os tipos de projetos, equipes e empresas pode ser contraditório aos princípios ágeis, além disso, é importante entender que um determinado

método não é a solução para todos os tipos de projetos e organizações. Outros aspectos importantes são: o surgimento, evolução e adequação dos métodos ágeis que podem ser apresentados ao longo dos anos. Podemos citar alguns métodos ágeis:

- Scrum.
- XP (*Extreme Programming*).
- FDD (*Feature Driven Development*).
- DSDM (*Dynamic Systems Development Method*).
- ASD (*Adaptative Software Development*)
- Crystal.

1.1 Situação-problema

As organizações escolhem os métodos ágeis de desenvolvimento de acordo com que acreditam ser mais apropriado, uma determinada metodologia escolhida pode ter sido selecionada baseando-se em um determinado aspecto, por exemplo, por se tratar de um método mais conhecido por gerentes, desenvolvedores ou por possuir maior quantidade de bibliografias ou mesmo por se tratar de um novo método ou sistema de desenvolvimento que implementa características mais atuais ao ambiente de desenvolvimento de software. Segundo Versionone (2013) o Scrum é um dos métodos ágeis mais utilizados, e o Kanban apresenta um crescimento de utilização bastante importante. Então surgiu o questionamento: Como o Scrum e o Kanban se comparam em termos de profundidade e abrangência no desenvolvimento ágil tomado como base o *Framework* SEMAT Essence?

1.2 Objetivos

OBJETIVO GERAL: Determinar a abrangência e profundidade do Scrum e do Kanban em relação ao *Framework* SEMAT *Essence*.

OBJETIVOS ESPECÍFICOS:

- Apresentar a metodologia SCRUM.
- Descrever os conceitos do sistema Kanban.
- Descrever o *Framework* SEMAT *Essence*.

- Analisar o método Scrum e o sistema Kanban de desenvolvimento em relação ao *framework* SEMAT *Essence*.

1.3 Hipóteses

- O Scrum e o Kanban podem apresentar similaridades e diferenças.
- A análise do Scrum e do Kanban em relação ao SEMAT pode contribuir para o entendimento de ambos quanto a sua utilização no desenvolvimento de software.

1.4 Justificativa

Existem diversos métodos ágeis que podem ser utilizados. A escolha do método ágil de desenvolvimento pode ser determinada por alguns fatores, e caso sejam escolhidos para um determinado contexto de desenvolvimento, ele pode não contribuir nesse processo de acordo com as características apresentadas pelo método, ou seja, pode se apresentar um risco para o projeto a ser desenvolvido maior do que um método tradicional.

Observou-se que o Scrum é um dos métodos mais utilizado e que atualmente o sistema Kanban para desenvolvimento de software surgiu como alternativa para ser utilizado no processo de desenvolvimento, dessa forma, surgiu o questionamento de como poderia ser comparado ambos e como poderia ter um embasamento a partir de um *Framework.*

A partir disso, surgiu a ideia de utilizar o *framework* SEMAT *Essence.* Também se acredita que a realização de estudos e pesquisas contribui para a geração de conhecimento sendo que o intuito é de proporcionar o entendimento e não o julgamento.

2 REVISÃO DE LITERATURA

2.1 Métodos Ágeis

Métodos ágeis possuem diversas metodologias e cada uma delas com suas características e particularidades, entretanto estão embasadas por princípios comuns, esses métodos buscam o trabalho cooperativo das equipes mais do que o formalismo e mais do que uma documentação escrita detalhadamente. Segundo Scrumguide.org (2015) em fevereiro de 2001 dezessete líderes de desenvolvimento de software cria o Manifesto para Desenvolvimento Ágil de Software, nesse manifesto Agilemanifesto.org (2001) descreve quatro valores:

- Indivíduos e interações, em relação a processos e ferramentas.
- Software funcionando em vez de documentação abrangente.
- Colaboração com o cliente em vez de negociação de contratos.
- Resposta à mudança em vez de seguir planos.

Segundo Pressman (2011, p.81) a metodologia ágil combina filosofia com um conjunto de princípios, onde a filosofia defende a satisfação do cliente, entrega incremental, equipes altamente motivadas e pequenas, artefatos mínimos e simplicidade no desenvolvimento.

> Geralmente, os métodos ágeis contam com uma abordagem interativa para especificação, desenvolvimento e entrega de software, e foram criados principalmente para apoiar o desenvolvimento de aplicações de negócio nas quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento. (SOMMERVILLE, 2009, p.262).

Os métodos ágeis foram desenvolvidos a partir do momento em que era necessário alternativas de desenvolvimento de software em determinados contextos onde foram identificados fraquezas nos processos convencionais da engenharia de software, a agilidade oferece diversos benefícios, mas não deve ser tratada com a solução de todos os problemas. Métodos ágeis segundo Pressman (2011, p.83) “[...]não é indicado para todos os projetos, produtos, pessoas e situações”.

O aspecto mais utilizado para embasamento para a utilização de métodos ágeis é a possibilidade que eles possuem em contribuírem para redução dos custos das mudanças ao longo do processo de desenvolvimento do software, incentivo a aplicação da filosofia ágil possibilitando uma melhor comunicação entre os envolvidos no projeto, entrega de forma incremental de funcionalidades do sistema, dimi-

nuição da importância de alguns artefatos de documentação e entendimento que o plano de projeto pode sofrer modificações.

2.2 Scrum

Segundo Scrumguide.org (2015) Jeff Sutherland e Ken Schwaber criaram o Scrum no início dos anos 90. Pressman (2013) descreve que o Scrum é utilizado para orientar o desenvolvimento de software em processos que possuem atividades: requisitos, análise, projeto, evolução e entrega.

Para Schwaber e Beedle (2013) Scrum é um framework dentro do quais pessoas podem tratar e resolver problemas complexos e adaptativos, realizando de forma produtiva e criativa entregas de produtos com o mais alto valor possível. O Scrum é leve, simples de entender e extremamente difícil de dominar.

> Scrum não é um processo ou uma técnica para construir produtos, entretanto é um framework dentro do qual podem ser empregados vários processos ou técnicas. Ele deixa claro a eficácia relativa das práticas de gerenciamento e desenvolvimento de produtos, de modo que possam ser melhorados, o Scrum consiste no time relacionado a papeis, eventos, artefatos e regras. Cada componente dentro do framework possui um propósito especifico e é essencial para o uso e sucesso do Scrum. (SCHWABER; BEEDLE. 2013).

Segundo Para Schwaber e Beedle (2013) o Scrum prega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos.

Mclaughlin (2015) descreve que o Scrum tem atraído crescente popularidade na comunidade de desenvolvimento de software ágil, devido à sua simplicidade, produtividade comprovada, e a capacidade de agir como uma base para diversas práticas de engenharia promovidas por outras metodologias ágeis.

Segundo Lindstrom (2015) organizações que adotaram o Scrum ter experimentado:

- Maior produtividade
- Produtos de melhor qualidade
- Redução de tempo de mercado
- Maior satisfação das partes interessadas

- Melhor dinâmica de equipe
- Funcionários mais satisfeitos

2.2.1 Papeis

Os times de desenvolvimento são auto organizáveis e multifuncionais. Times auto organizáveis escolhem qual a melhor forma para complementarem seu trabalho, em vez de serem dirigidos por outros de fora do time. O modelo de time no Scrum é projetado para aperfeiçoar a flexibilidade, criatividade e produtividade. Schwaber e Beedle (2013) apresentam três papeis para o Scrum:

- *Product Owner:* representa o cliente, sendo responsável por gerenciar o backlog do produto.
- *Scrum Master:* é responsável por garantir que o Scrum seja entendido e aplicado, busca a garantia de que o time adere às teorias, práticas e regras do Scrum.
- *Times de desenvolvimento:* são os profissionais responsáveis pelo desenvolvimento do produto, são os únicos que criam incrementos, os times são estruturados e autorizados pela organização e gerenciamento de seu próprio trabalho.

2.2.2 Eventos

Segundo Schwaber e Beedle (2013) eventos são usados no Scrum para criar uma rotina e minimizar a necessidade de reuniões não definidas, todo evento possui uma duração máxima, é uma oportunidade de inspeções e adaptações.

Sprint é uma versão incremental potencialmente utilizável do produto, possui duração coerente em todo o processo de desenvolvimento e um novo *Sprint* é iniciado imediatamente após a conclusão do Sprint anterior.

> Segundo Schwaber e Beedle (2013) durante o Sprint não são realizadas mudanças que possam adicionar risco no objetivo do Sprint; As metas de qualidade não diminuem e o escopo pode ser apresentado e renegociado entre o dono do produto e o time de desenvolvimento. Os Sprints são limita-

> dos no período de um mês, assim podem permitir previsibilidade a cada mês corrido, além de limitar o risco ao custo.

Os *Sprints* são compostos por reunião de planejamento do *Sprint*, reunião diária, trabalho de desenvolvimento, reunião de revisão e retrospectiva do *Sprint*. Segundo Schwaber e Beedle (2013) é descrito os seguintes pontos:

- *Planejamento do Sprint:* o time determina uma previsão das funcionalidades que serão desenvolvidas durante o *Sprint*, selecionando os itens do *backlog* do produto e a determinação dos objetivos do time, sendo o próximo passo decidir como essas funcionalidades serão. Segundo KNIBERG (2008) escopo e importância dos itens são definidos pelo *product owner* e a estimativa é determinada pela equipe.

- *Reunião diária:* é realizada para inspecionar o trabalho desde a última reunião diária e prever o trabalho que deverá ser realizado antes da próxima, sua duração é de 15 minutos, observa-se o andamento do trabalho. O Scrum Master assegura que a reunião seja realizada e que somente o time participe, mas o time é que conduz a mesma, ela melhora a comunicação, elimina outras reuniões, identifica e remove impedimentos para o desenvolvimento, destaca e promover rápidas tomadas de decisão e melhora o nível de conhecimento do time.

- *Revisão da Sprint:* nesse evento é realizado de inspeção do incremento e a adaptação do Backlog do produto se necessário, ela é executada no final do Sprint. O resultado dessa reunião é um *backlog* do produto revisado que define o provável *Backlog* do produto para o próximo *Sprint*. O *backlog* do produto pode também ser ajustado completamente para atender novas oportunidades.
- *Retrospectiva da Sprint*: é uma oportunidade para o time inspecionar o próprio time e criar um plano para melhorias a serem aplicadas no próximo *Sprint*.

2.2.3 Artefatos

Segundo Schwaber e Beedle (2013) os artefatos do Scrum representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação, os artefatos:

- *Backlog do Produto:* É uma lista ordenada de tudo que deve ser necessário no produto, é uma origem única dos requisitos para qualquer mudança a ser realizada, *o Product Owner* é o responsável, incluindo conteúdo, disponibilidade e ordenação do *backlog,* inicialmente são estabelecidos os requisitos conhecidos e entendidos, ele é dinâmico mudando constantemente para identificar o que o produto necessita para ser mais apropriado, competitivo e útil. Segundo Pressman (2011, p.95) *backlog* é uma lista com prioridade dos requisitos ou funcionalidades do projeto que fornecem valor comercial ao cliente. Os itens podem ser adicionados a esse registro em qualquer momento.

- *Backlog do Sprint:* É um conjunto de itens do *backlog* do produto selecionados para o Sprint, é a previsão do time de desenvolvimento sobre qual funcionalidade estará no próximo incremento e sobre o trabalho necessário para entregar uma funcionalidade em um incremento “Pronto”. Somente o time de desenvolvimento pode alterar o *backlog* do *Sprint* durante o *Sprint*, além de monitorar o total do trabalho restante pelo menos a cada reunião diária, o time acompanha estes resumos diários e projeta a probabilidade de alcance dos objetivos do *Sprint*.

2.3 Kanban

O Kanban é uma palavra de origem japonesa e seu significado pode ser considerado “Cartão visual”, ele é conhecido por ser utilizado por mais de 50 anos no sistema de produção da Toyota. O pioneiro no uso do Kanban para o desenvolvimento de software foi David J. Anderson. Segundo Klipp (2015) o Kanban é uma das ferramentas do Lean mais populares e utilizadas.

O Kanban é inspirado pelo trabalho de Don Reinertesn sobre o desenvolvimento *Lean* de Produtos. Esses conceitos aplicados no processo de desenvolvimento de software permitem observar que na necessidade de implementar uma nova funcionalidade para um sistema, essa demanda somente será realizada quando uma funcionalidade anterior já tenha sido implementada. O Kanban busca o aperfeiçoamento dos processos, equipes e projetos, permitindo a melhoria contínua, aumento de produtividade e a melhora na relação com os clientes. Boeg (2012) descreve que os princípios do Kanban são:

- Visualizar o trabalho em andamento;
- Visualizar cada passo em sua cadeia de valor;
- Limitar o trabalho em progresso, *WIP - Work in Progress;*
- Tornar explícita as politicas sendo seguidas.
- Medir e gerenciar o fluxo;
- Identificar oportunidades de melhorias, na qual a melhoria contínua é responsabilidade de todos.

> Para Boeg o sistema Kanban "[...] representa uma implementação mais direta dos princípios de desenvolvimento Lean de produtos para o desenvolvimento de software que os métodos ágeis tradicionais. Com foco consistente no fluxo e no contexto, o Kanban oferece uma abordagem menos prescritiva comparada ao Agile, e tem se tornado uma extensão popular dos métodos ágeis tradicionais como Scrum e XP." (BOEG, 2012).

Segundo Mclaughlin (2015) o Kanban é utilizado pelas organizações para gerenciar a criação de produtos com ênfase na entrega contínua enquanto não sobrecarrega a equipe de desenvolvimento. Como Scrum, o Kanban é um processo concebido para ajudar as equipes a trabalharem juntos de forma mais eficaz. Além disso, o Kanban promove a colaboração contínua, incentiva à aprendizagem contínua e definindo o melhor fluxo possível de trabalho para a equipe.

Através do quadro de atividades a equipe pode observar a quantidade de esforço que poderá ser necessário para executar as demandas, levando-se em consideração o limite de capacidade da equipe e do sistema que está sendo desenvolvido.

Um dos principais objetivos do Kanban é avaliar o trabalho em progresso chamado WIP.

> Boeg descreve o wip: “[...] O WIP apresenta o total de trabalho ou atividade em progresso no sistema Kanban, dependendo do contexto de sua utilização. A atividade ou tarefa pode ser considerado no WIP a partir do momento em que se encontram no backlog, entretanto em um outro contexto essas mesma atividades ou tarefas podem ser considerados no WIP a partir da iniciação de execução".(BOEG, 2012).

Segundo Boeg (2012) “[...] visualizar o fluxo e estabelecer os limites de *WIP*, garantirá que nunca se pode introduzir mais trabalho no sistema que a capacidade do sistema de processar esse trabalho.”.

Para Klipp (2015) existem três aspectos básicos para a implementação do Kanban:

- Visualizar o trabalho.
- Limitar o trabalho em progresso.
- Medir e melhorar o processo.

Sobre o Kanban, Klipp (2015) descreve que sua adoção não significa que não seja necessário rever os métodos e processos já existentes. Depois de mapear o fluxo de trabalho pode-se construir o quadro Kanban, este quadro é uma tabela que possui colunas relacionadas a cada passo no fluxo de trabalho. Após a criação do quadro pode se adicionar as tarefas que serão executadas.

Para Klipp (2015) certas vantagens são apresentadas simplesmente como resultados da visualização do fluxo do trabalho, mas outras necessitam de mais esforço. Duas ferramentas poderosas para acompanhar as melhorias e conhecimento são o Lead time e *Cycle time*.

- *Lead time* é basicamente quanto tempo leva para que uma tarefa a partir do momento que é solicitada ate o momento de sua entrega.
- O *Cycle Time* é o tempo que um membro do time leva para finalizar uma tarefa, a partir do momento de seu inicio.

2.4 SEMAT

Segundo SEMAT (2013) SEMAT (Software Enginnering Method and Teory) é uma iniciativa lançada no fim de 2009 por Ivar Jacobson, juntamente com outros dois conselheiros SEMAT, Bertand Meyer e Richard Soley.

Segundo Ivar Jacobson International (2015a), o SEMAT é uma comunidade da indústria empenhada em melhorar as práticas de engenharia de software através da redefinição da engenharia de software como uma disciplina rigorosa. Um dos primeiros e importante resultado da comunidade SEMAT é o *Essence*, um *framework* que define um pequeno conjunto de conceitos que são comuns em todos os projetos de software, ajudando equipes a avaliarem seus esforços e melhorarem sua forma de trabalho.

Segundo SEMAT (2015b) a engenharia de software é prejudicada todos os dias por práticas imaturas, o SEMAT pretende reformular os aspectos relacionados aos métodos da engenharia de software baseado em uma teoria sólida, princípios comprovados e melhores práticas que:

- Inclui um *Kernel* de elementos amplamente comprovados, extensível para usos específicos;
- Aborda tanto aspectos de tecnologia quanto de pessoas;
- É apoiado pela indústria, academia, pesquisadores e usuários;
- Extensão de suporte em face de evolução dos requisitos e tecnologia;

2.4.1 Kernel

Para abordar as práticas comuns foi desenvolvido o *Kernel,* ele é um conjunto de elementos essenciais utilizados para criação de uma base comum que descreva o esforço na engenharia de software. Os elementos do *Kernel* possuem estados que representam o andamento do processo de desenvolvimento, entre outras coisas ajudam os profissionais a comparar métodos e a tomar a melhor decisão sobre suas práticas.

Os elementos do *Kernel* formam uma base sobre o qual se pode definir e descrever qualquer método ou prática existente ou futura, esses elementos são defi-

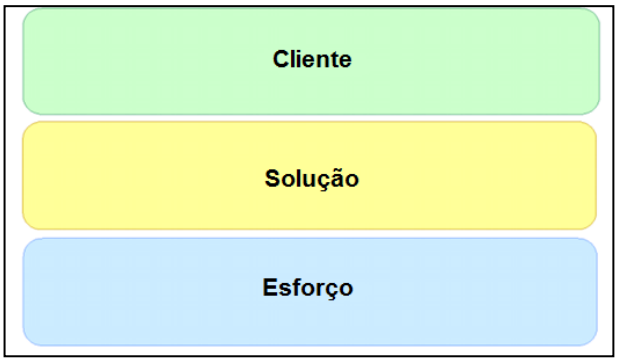
nidos de uma forma que sejam extensíveis e adequáveis, suportando uma ampla variedade de práticas, métodos e estilos de desenvolvimento mesmo que o ambiente de desenvolvimento não possua um método definido, o *Essence Kernel* pode ser utilizado para monitorar o andamento do trabalho desenvolvido pelas equipes e permite a análise dos pontos fortes e fracos desse trabalho.

O foco do *Kernel* é definir uma base comum para a definição de práticas de desenvolvimento, que permitam que essas práticas sejam aplicadas de forma independentemente do projeto. As práticas podem ser combinadas para criarem métodos de engenharia de software específicos adaptados às necessidades específicas de uma comunidade de engenharia de software, projeto, equipe ou organização.

Segundo o SEMAT (2013) o *Kernel* é organizado em três áreas de interesse, cada uma focada a um aspecto específico da engenharia de software:

- *Cliente:* Está área de interesse contém tudo a ver com o uso atual e exploração do sistema de software que será produzido.
- *Solução:* Está área de interesse contém tudo a ver com a especificação e desenvolvimento do sistema de software.
- *Esforço:* Está área de interesse contém tudo a ver com a equipe e de que maneira ela realiza o seu trabalho.

Figura 1: As três Áreas de Interesse



Fonte: Tradução do Guia SEMAT

Cada área de interesse contém um pequeno número de *alphas*, espaço de atividades e competências:

- *Alphas (Abstract-Level Progress Health Attribute):* são representações essenciais que serão realizadas. Os alphas fornecem descrições do que uma equi-

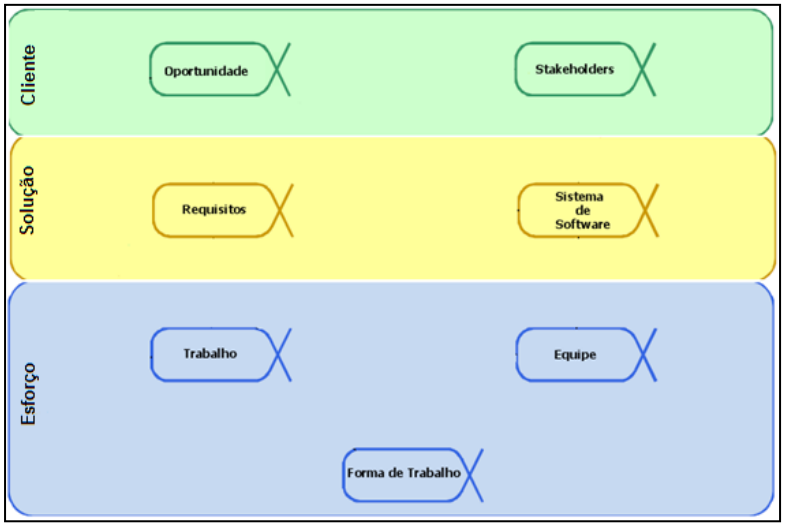
pe irá administrar, produzir e utilizar durante o processo de desenvolvimento, manutenção e suporte do software.

- *Espaço de Atividades (Activity Spaces):* são representações essenciais do que serão realizados, providencia descrições dos desafios que a equipe enfrentará no desenvolvimento, manutenção e suporte dos sistemas e o que foi feito para superá-los durante o desenvolvimento.
- *Competências (Competencies):* são representações das competências chaves necessárias para a implementação do software.

2.4.2 Alphas

O *Kernel* possui sete *alphas* que são distribuídos entre as áreas de interesse cliente, solução e esforço:

Figura 2: *Alphas* por área de interesse

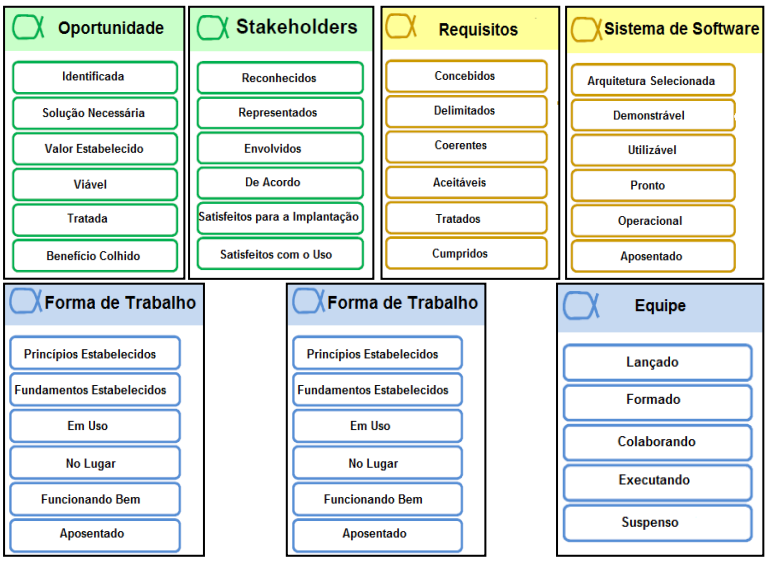


Fonte: Tradução do Guia SEMAT

Na área de interesse clientes temos a oportunidade e os *stakeholders*, na área de interesse solução temos o os requisitos e o sistema de software e no esforço temos o time, forma de trabalho e o trabalho.

Cada *alpha* possui um pequeno conjunto de estados pré-definidos que são usados para avaliação do desenvolvimento. Os *alphas* representam indicadores críticos dos aspectos mais importantes para serem monitorados e acompanhados.

Figura 3: Estados de cada Alpha



Fonte: Tradução do Guia SEMAT

São propostos cartões para representá-los, nestes cartões são apresentados os nomes dos *alphas*, os critérios para aquele estado e o número do estado relacionado a um total de estados. Na Figura 4, um exemplo de cartões do *Alpha* Requisitos:

Figura 4: Cartões do Alpha Requisitos

Fonte: Tradução do Guia SEMAT

Segundo Ivar Jacobson International (2015a), os cartões são uma maneira simples, fácil de controlar o andamento de um projeto de software e ajuda no planejamento dos próximos passos. Os estados dos *alphas* representam diferentes fases de progresso de um projeto de desenvolvimento de software, e cada estado é alcançado quando os critérios indicados de cada estado no cartão são atendidos.

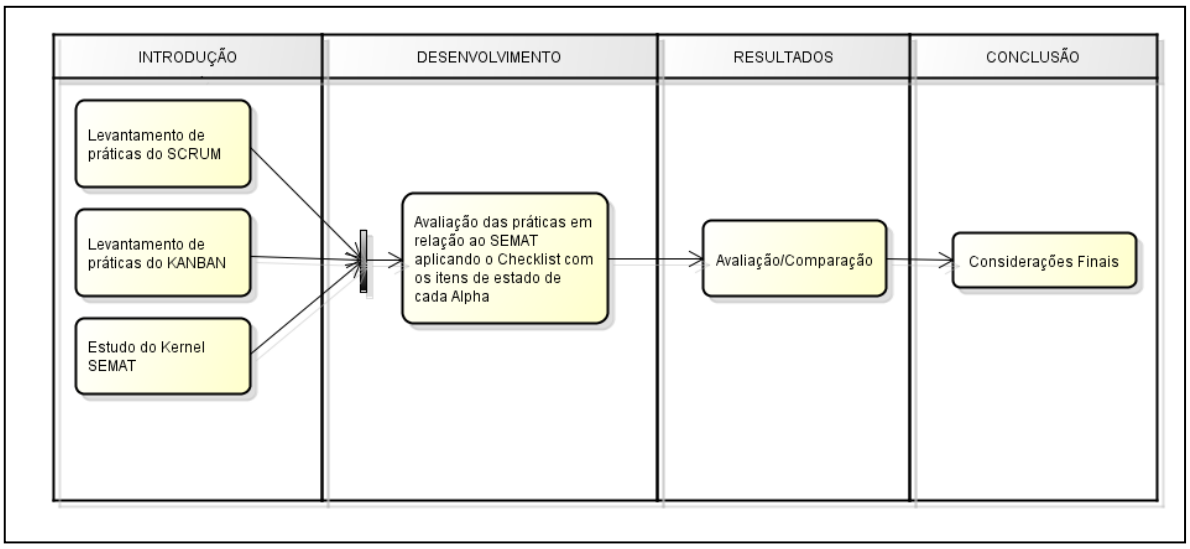
Os cartões de estados *alphas* podem contribuir na compreensão do estado atual do projeto, solucionar problemas específicos de uma área, definir os objetivos para iterações futuras, facilitar retrospectivas e indicar as melhorias em cada área.

3 METODOLOGIA

O desenvolvimento desse estudo foi realizado com base em pesquisa descritiva, sendo que a pesquisa aplicada foi qualitativa. Buscando responder ao questionamento que é o incentivador para o desenvolvimento desse trabalho, apresentar quais os aspectos resultantes da comparação do Scrum e do sistema Kanban tomando como critérios de comparação os elementos do *Framework* SEMAT *Essence*. A metodologia utilizada foi definida por alguns passos, sendo inicialmente destacada pela breve síntese do método Scrum, os aspectos do sistema Kanban, além da apresentação do *Framework* SEMAT *Essence*, com seus elementos e suas práticas.

Posteriormente foi realizada a avaliação das práticas destacadas para cada objeto de pesquisa e foram analisados em relação ao SEMAT para verificar os aspectos comuns ao estabelecido no SEMAT, ou seja, os *Alphas* e estados. E finalmente apresentar o observado na análise estudada entre o Scrum e o Kanban e buscar o entendimento dessa análise para contribuição na geração de conhecimento e incentivar mais estudos teóricos e práticos na engenharia de software. A Figura 5 apresenta ...

Figura 5: Diagrama de Atividades da Metodologia desse Trabalho



Fonte: Elaborado pelo Autor

4 DESENVOLVIMENTO

Inicialmente foram apresentados de forma resumida conceitos do Scrum, Kanban e o SEMAT. Observou-se que o SEMAT possui sete *alphas*, sendo que cada um deles possui estados, cada estado possui seu *checklist* de critérios. Dessa forma, para o desenvolvimento do trabalho foi avaliado o *checklist* de cada estado no Scrum e no Kanban para possibilitar a análise comparativa, observando se cada metodologia possui conceitos ou práticas que possam atender a cada critério do *checklist*, procurando avaliar se cada critério foi atendido, não atendido ou provavelmente atendido. Além disso, a possibilidade de não existir a definição de atendido ou não atendido foi considerada no trabalho, definido como não se aplica, por questões subjetivas relacionadas a uma aplicação de um projeto real e que tais critérios poderiam ser melhor comparados, considerando as prescrições das prática de modo teórico para as práticas do Scrum e Kanban. As tabelas 1 e 2 podem exemplificar a análise realizada no desenvolvimento do trabalho.

Tabela 1: Checklist do Alpha Requisitos do Scrum

2 - Delimitados	Critério	Justificativa
Os stakeholders envolvidos no desenvolvimento do novo sistema foram identificados	Atendido	Reunião de Planejamento de Versão para Entrega e Product Owner
Os stakeholders estão de acordo com o objetivo do novo sistema.	Provavelmente Atende	Product Owner
Está claro o significado de sucesso para o novo sistema.	Atendido	Product Backlog
Os stakeholders compartilham a compreensão sobre o alcance da solução proposta.	Atendido	Product Owner; Reunião de Planejamento de Sprint
Há concordância sobre a forma de descrição dos requisitos.	Atendido	Backlog do Produto; Backlog da Sprint
Mecanismos para gerenciar os requisitos foram instalados.	Atendido	Product Backlog; O Scrum Master contribui para que o Product Owner encontre técnicas para o gerenciamento efetivo do Backlog do Produto.
O esquema de priorização é claro.	Atendido	Product Backlog; Product Owner responsável por deixar claro os itens do backlog; Backlog da Sprint
Restrições foram identificadas e consideradas.	Provavelmente Atende	Reunião de Planejamento de Versão para Entrega
Premissas foram claramente descritas.	Não Atende	

Fonte: Elaborado pelo autor

Na tabela 1, pode-se visualizar que o Scrum atende ou provavelmente atende a maior parte dos critérios do estado delimitados para o *alpha* requisitos, justificado pelos conceitos de *Product Owner, Product Backlog* e Reunião de Planejamento.

Tabela 2: Checklist do Alpha Requisitos do Kanban

2 - Delimitados	Critério	Justificativa
Os stakeholders envolvidos no desenvolvimento do novo sistema foram identificados	Não se Aplica	
Os stakeholders estão de acordo com o objetivo do novo sistema.	Não se Aplica	
Está claro o significado de sucesso para o novo sistema.	Provavelmente Atende	Estabelecer Politicas para garantia de qualidade
Os stakeholders compartilham a compreensão sobre o alcance da solução proposta.	Não se Aplica	
Há concordância sobre a forma de descrição dos requisitos.	Não se Aplica	
Mecanismos para gerenciar os requisitos foram instalados.	Provavelmente Atende	Backlog de itens e Product Backlog
O esquema de priorização é claro.	Provavelmente Atende	Priorização opcional
Restrições foram identificadas e consideradas.	Atendido	Estabelecer Politicas para garantia de qualidade
Premissas foram claramente descritas.	Não se Aplica	

Fonte: Elaborado pelo autor

Na tabela 2, pode-se observar que o Kanban não se aplica grande parte dos critérios do estado delimitados, de acordo com o examinado, o Kanban não prescreve muitas práticas relacionadas ao alpha requisitos, entretanto considerar que não são atendidos depende do contexto do projeto e especificamente de projetos reais, justificando a quantidade significativa de “não se aplica” em contraposição ao “não atendido”.

Os *checklists* elaborados no desenvolvimento do trabalho podem ser acessado em : https://www.dropbox.com/sh/unwmzjsdyj4yfkr/AABOrloSQAXMGMEPZmKcW99fa?dl=0

5 RESULTADOS

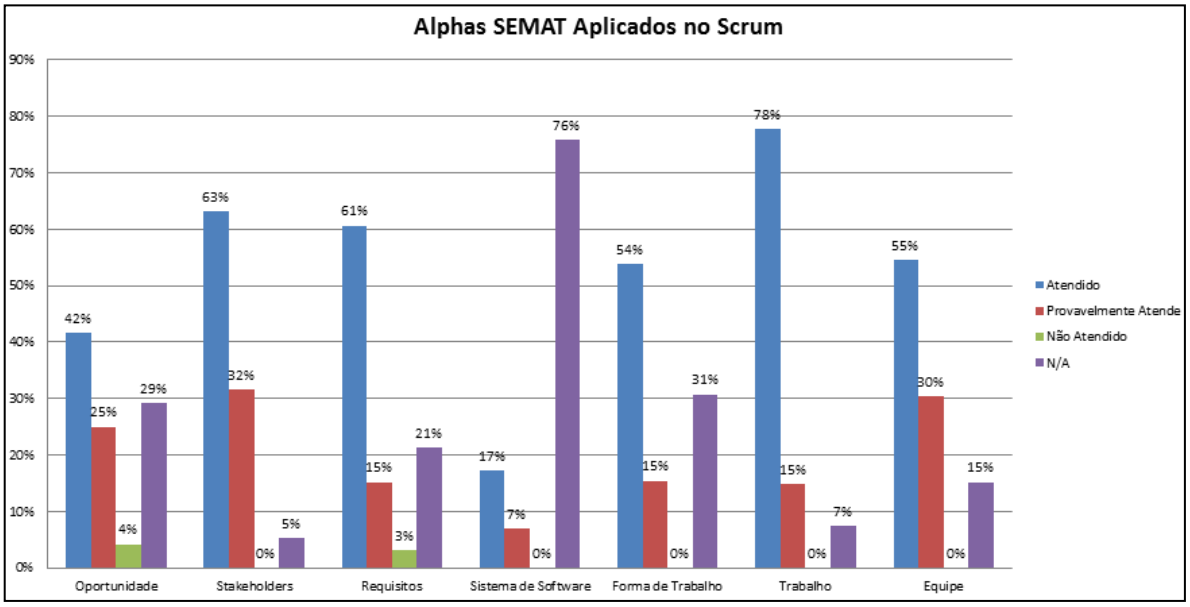
Durante a pesquisa alguns aspectos foram observados e de forma interessante com relação a semelhança e diferenças entre o Kanban e o Scrum, por exemplo, O Scrum prescreve papeis em contrapartida o Kanban não prescreve, o Kanban limita as tarefas que está em execução num determinado ponto do processo por estado de fluxo de trabalho e o Scrum por iteração.

Ambos são empíricos, ou seja, colocando seus princípios de forma prática e adaptando ao contexto do projeto e organização, além disso possuem os princípios de sistema puxado, melhoria contínua e resposta a mudanças ao invés de planos pré-estabelecidos

Eles limitam a atividade em andamento, realizam o trabalho de forma dividida em parte, além de entrega rápida de software que funcione e equipes auto organizáveis.

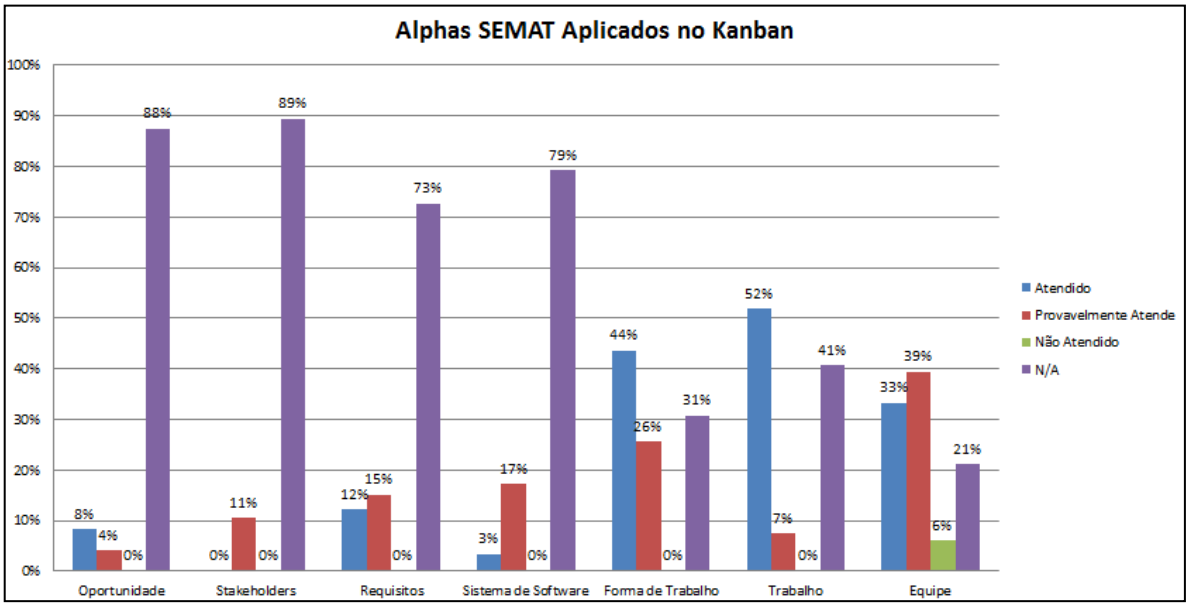
Foram gerados dois gráficos para a análise e comparação, utilizando se as classificações de critérios: Atendido, Provavelmente Atende, Não Atende e Não se Aplica (N/A). Nos gráficos foram utilizados indicadores em percentual para cada classificação de acordo com cada *alpha.*

Figura 6: Gráfico de Aplicação do SEMAT no Scrum



Fonte: Elaborado pelo autor

Figura 7: Gráfico de Aplicação do SEMAT no Kanban



Fonte: Elaborado pelo autor

Pode-se observar que o Scrum se diferenciou do Kanban principalmente nos *Alphas:* Oportunidade, *Stakeholders,* Requisitos e Trabalho. Nestes o Scrum atendeu os critérios em percentual acima de 50%, sendo o alpha Oportunidade o único abaixo desse percentual, com 42% de atendimento. Já o Kanban obteve um percentual acima de 70% de critérios não aplicados, exceto o alpha trabalho que obteve apenas

41% de não aplicados, entretanto seu percentual de atendimento foi de 52% contra 78% do Scrum.

O alpha Sistema de Software foi o alpha em que o Scrum e o Kanban mais se aproximaram, principalmente pelos critérios do alfas com uma um percentual equivalente de não aplicados sendo 76% no Scrum e 79% no Kanban.

Durante a análise, observamos que tanto o Scrum quanto o Kanban não prescreve explicitamente práticas ou conceitos relacionados aos aspectos técnicos da construção do software propriamente dita uma vez que são métodos mais voltados para o controle das tarefas e da equipe.

Os *alphas* Forma de Trabalho e Equipe tiveram percentuais próximos, entretanto o Scrum obteve maiores percentuais de atendimento nos dois estados, na Forma de Trabalho o Scrum obteve 54% de atendimento contra 44% do Kanban e no *alpha* Equipe, o Scrum obteve 55% de atendimento contra 33% do Kanban.

Outro informação importante são os percentuais de prováveis critérios atendidos no Scrum nos *alphas:* Oportunidade, *Stakeholders* e Equipe que obtiveram 25%, 32% e 30% respectivamente. Além disso, no Kanban o alpha equipe obteve 6% de critério não atendido.

De acordo com os critério utilizados para os *alphas* do SEMAT, entende-se que o Kanban não deve ser utilizado separadamente como método de desenvolvimento de software pois não engloba alguns aspectos importantes da engenharia de software, dessa forma, pode ser sugerido que ele deve ser utilizado em conjunto com outro método, além disso, ele não deve ser considerado um substituto para o Scrum, sendo que podem ser utilizados em conjunto, sendo o Scrum como o ponto de partida na definição de um método de desenvolvimento e o Kanban colaborando com sua práticas, sendo adaptados para cada contexto de um projeto. O Scrum amplia as partes da engenharia de software de forma mais madura em comparação ao Kanban.

6 CONCLUSÃO

O trabalho apresentou os conceitos e práticas do método Scrum, o entendimento do sistema Kanban para desenvolvimento de software e descreveu de forma objetiva e sinteticamente o *Framework* SEMAT. Além disso, apresentou a análise com-

parativa entre o Scrum e o Kanban considerando a cobertura dos critérios para os estados de cada alpha do SEMAT utilizando-se de um checklist.

Considerando os dados gerados durante o desenvolvimento do trabalho e apresentando os dados graficamente para melhor compreensão da análise comparativa, pode-se concluir que o Scrum é mais prescritivo que o Kanban pois apresenta uma maior quantidade de regras, sendo o Kanban mais adaptativo.

Visualizando os gráficos gerados, conclui-se que o Scrum atende uma quantidade significativa de critérios em relação ao Kanban, o Kanban por ser mais adaptativo não permite descrever com precisão se determinado critério é atendido, entretanto não permite apontar de forma consistente que um determinado critério não possa ser atendido, induzindo ao entendimento que o atendimento vai depender do contexto do projeto e da organização.

Foi observado que o Scrum e o Kanban podem ser utilizados em conjunto, adaptando-se a realidade do projeto e mesclando suas práticas e princípios. O Scrum se diferenciou do Kanban principalmente nos alphas: Oportunidade, Stakeholders, Requisitos e Trabalho.

6.1 Trabalhos Futuros

Existem diversas possibilidades de aprofundamento e estudos relacionados ao métodos ágeis e ao SEMAT, a produção de artigos e trabalhos através de comparações de métodos contribui para a geração de conhecimento para os projetos de desenvolvimento de software, outro viés importante é a geração do conhecimento através de estudos de casos de projetos reais, demonstrando os resultados práticos das aplicações do conhecimento teórico dos métodos e a aplicação do Framework SEMAT.

REFERÊNCIAS

AGILEMANIFESTO.ORG. **Manifesto for Agile Software Development.** 2001. Disponível em: < http://agilemanifesto.org/ >. Acesso em: 22 abril. 2015.

ANDERSON J. David. **Kanban com o pioneiro: Entrevista com David J. Anderson.** 2012. Disponível em: < http://www.infoq.com/br/articles/kanban-david-anderson-conceitos-e-mitos >**.** Acesso em: 23 abril. 2015.

BOEG, Jesper. **Kanban em 10 Passos.** 2012. Disponível em: <http://www.infoq.com/br/minibooks/priming-kanban-jesper-boeg>. Acesso em: 23 dez. 2014.

BRIA, Mike**. Comparando Kanban a Scrum.** 2009. Disponível em: <http://www.infoq.com/br/news/2009/05/kniberg-kanban-v-scrum>. Acesso em: 21 dez. 2014.

IVAR JACOBSON INTERNATIONAL. **Alpha State Cards: A Lean and Lightweight Governance Approach for Agile Software Development.** Disponível em: <http://www.ivarjacobson.com/alphastatecards**/>.** Acesso em: 27 abril. 2014a.

IVAR JACOBSON INTERNATIONAL. **bSoftware Engineering Method and Theory (SEMAT).** Disponível em: < http://www.ivarjacobson.com/Software_Engineering_Method_and_Theory/**/>.** Acesso em: 27 abril. 2014b.

KLIPP, Paul. **Getting Started With Kanban.** Disponível em: https://kanbanery.com/ebook/GettingStartedWithKanban.pdf**>.** Acesso em: 23 dez. 2014.

KNIBERG, Henrik; SHARIN, Mattias.**Kanban e Scrum - obtendo o melhor de ambos.** 2010**.** Disponível em:<http://www.infoq.com/br/minibooks/kanban-scrum-minibook**>.** Acesso em: 21 dez. 2014.

KNIBERG, Henrik. **Scrum e XP direto das Trincheiras.** 2008. Disponível em: <http://www.infoq.com/br/minibooks/scrum-xp-from-the-trenches**>.** Acesso em: 23 dez. 2014.

LINDSTROM, Lowell. **What Is Scrum?.** Disponível em: < http://www.versionone.com/agile-101/what-is-scrum/ >. Acesso em: 22 abril. 2015

MCLAUGHLIN, Mike. **Agile Methodologies for Software Development.** Disponível em: < http://www.versionone.com/agile-101/agile-development-methodologies-scrum-kanban-lean-xp/ >. Acesso em: 22 abril. 2015.

PRESSMAN, Roger S. **Engenharia de software.** Rio de Janeiro: McGraw-Hill, 2011.

SCHWABER, K; BEEDLE, M. **Um guia definitivo para o Scrum: As regras do jogo. 2013.** Disponível em: < http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf **>.** Acesso em: 22 abril. 2015.

SCRUMGUIDE.ORG. **The History of Scrum.** Disponível em: < http://www.scrumguides.org/history.html **>.** Acesso em: 22 abril. 2015.

SEMAT. **Kernel and Language for Software Engineering Methods (Essence). 2013. Beta 1.** Disponível em:< http://www.omg.org/spec/Essence/1.0/PDF/**>.** Acesso em: 22 dez. 2014.

SEMAT. **Call for Action.** Disponível em:< http://semat.org/?page_id=2**>.** Acesso em: 28 abril. 2015.

SOMMERVILLE, Ian. **Engenharia de software**. São Paulo: Addison-Wesley, 2003.

VERSIONONE Agile Made Easier. **8th Annual State Of Agile Survey.** Disponível em: <http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf**>.** Acesso em: 22 dez. 2014.