ENTENDENDO O DESENVOLVIMENTO ÁGIL

Luis Augusto Trindade Curado
Giselle Barbosa Gomes Machado
Rogério Oliveira da Silva

Resumo: Desenvolver softwares é uma atividade difícil e ariscada. Entre os maiores riscos estão: o orçamento, consumo de tempo abusivo, funcionalidade que não supri as necessidades do cliente, baixa qualidade dos sistemas desenvolvidos e por fim cancelamento por inviabilidade. O desenvolvimento ágil foi criado de forma independendo por diversos profissionais renomados na área de engenharia de software, só conseguindo minimizar os riscos associados ao desenvolvimento de software, pensando de forma muito diferente do que tradicionalmente. Estes profissionais, decidiram reunir-se para criarem o Manifesto Ágil de Software

Palavras-chaves: metodologias ágeis, agilidade em desenvolvimento de software

Abstract: *Developing software is a difficult and challenging activity. Among the biggest risks are: the budget, consumption of time abusive, functionality that did not meet the needs of the client, low quality of the developed systems and finally cancellation by non-viability. Agile development was created independently by several renowned professionals in the area of software engineering, only managing to minimize the risks associated with software development, thinking in a very different way than traditionally. These professionals decided to meet to create the Agile Software Manifesto*

Keywords: *agile methodologies, agility in software development*

Introdução

Hoje um dos grandes vilões para o desenvolvimento de software é o tempo, resumindo os clientes querem entregas o mais rápido possível, querem poder ver seus investimentos, testar. As metodologias tradicionais, tais como a Cascata não permite que o cliente veja o software em um curto período de tempo, pois as metodologias tradicionais seguem o passo a passo, primeiro fazendo o levantamento de requisitos e análise, logo o design UML só depois desse processo todo que finalmente começa o desenvolvimento com isso é perdido muito tempo e é comum o cliente desistir do projeto ou até mesmo na entrega não ser mais funcional para o cliente.

As metodologias Ágil é focada na entrega do produto, nem que seja por partes para que com isso o cliente possa ver o que está sendo desenvolvido e se ele quer continuar com o projeto, do mesmo modo o ágil é muito mais flexível para que haja mudança de escopo sempre agregando valor ao projeto. A metodologia ágil faz o uso de vario processos, tais como Scrum, XP, Kanban etc. com aumento significativo de adeptos ao ágil surgiu o Manifesto Ágil no qual é composto por quatro valores e diversos princípios, para com isso poder nortear os seguidores do desenvolvimento ágil.

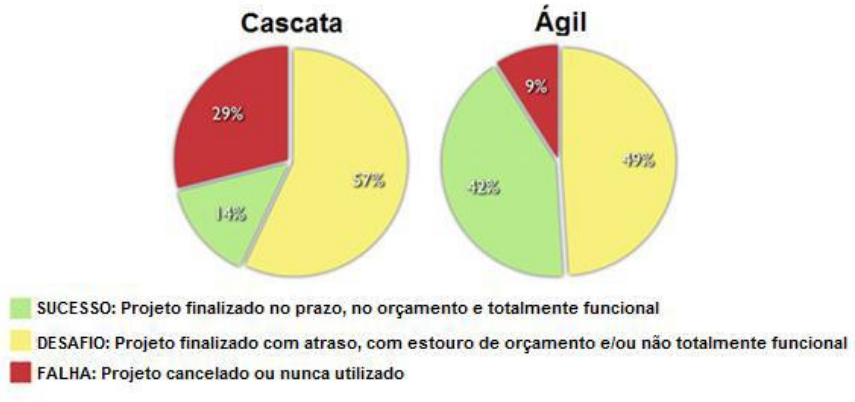

Gráfico 1: Comparativo do sucesso dos projetos que seguem a metodologia em Cascata versus a metodologia Ágil (Adaptado de STANDISH 2012).

Métodos Tradicionais

Problemas com os métodos tradicionais, tais como o Waterfall (figura 1.1) ou simplesmente Cascata. O termo foi originado em um artigo publicado em 1970 por W.W. Royce. O próprio autor descreveu o método como arriscado a propenso a falhas, no qual havia diversas fases de um projeto, onde era feito um levantamento de requisito muito bem detalhado de todas as funcionalidades do sistema, isso era necessário para seguir as exigências do contrato, mantido por ambas as partes, no qual era conhecido como contrato de escopo fechado.

Na fase de levantamento de requisito logo no início do projeto era fechado todo o escopo do projeto, assim o cliente precisava validar os requisitos do projeto logo no início do processo, muito embora ele não tenha tido a oportunidade de ver como o projeto vai ser executado e se o que ele decidiu no início do projeto vai mesmo ser necessário para as demais fases, com isso o cliente tinha que tomar muitas decisões críticas e ariscadas definir se o projeto seria um fracasso ou um sucesso. Depois, seguia para uma nova etapa, de análise de software, em que, uma equipe fazia a análise de todos esses requisitos e cada etapa desse processo costuma demorar muito tempo. Por exemplo, cada passo teria vários meses de duração.

Após o processo de análise, passava-se para outra equipe que fazia o design, modelagem UML (Unifiel Modeling Language). Após um bom tempo, iniciava-se a codificação, onde uma equipe era responsável pela codificação e outro pelos testes, quando faziam. Pelo fato dos testes estarem no fim do processo, muitas das vezes eram deixados de lado por conta do curto prazo determino no escopo do projeto, onde só eram realizados os testes se sobrasse tempo e isso acabava gerando muitos problemas de qualidade no produto a ser entregue.

A metodologia Waterfall são várias fazes bem definidas, geralmente realizada por pessoas com skills diferentes, por equipe, inclusive, distintas. Nessa lógica, apenas quando uma etapa estiver completa, passa-se para a próxima, a ideia era que as etapas avançassem linearmente para a próxima. O grande problema dessa abordagem é que quando o cliente obtinha o produto final, ele manifestava descontentamento, “Não foi isso que eu pedi”, ”Não era isso que eu precisava” e “Isso não me atende”.

Se por algum motivo o cliente precisasse para um projeto no meio, não haveria nada funcionando, nenhum resultado, uma vez que o produto só ficava pronto no final do projeto. Portanto o cliente recebia o retorno do investimento apenas ao fim do projeto como um todo. Por esse e outros motivos uma grande parte dos projetos acaba sendo interrompido antes do

fim, isso ocorre por N motivos, o produto não é mais necessário, a desconfiança do cliente para com a o processo moroso e outros motivos.

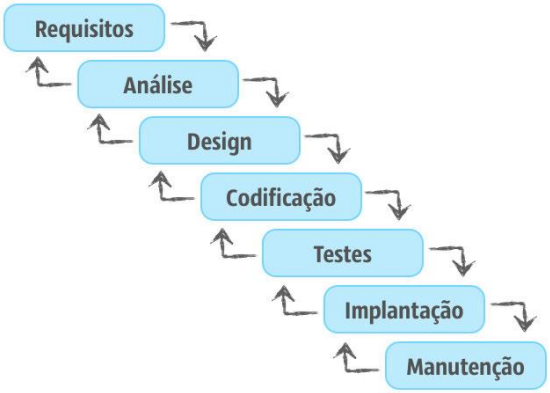

Figura 1.1: Processo em Cascata (Waterfall)

Metodologias ágeis

"Não é o mais forte que sobrevive, nem o mais inteligente, mas o que melhor se adapta as mudanças."

– Charles Darwin

Nos últimos anos, os métodos ágeis vêm ganhando mais popularidade, e grandes empresas como Google, Yahoo!, Microsoft, IBM, Cisco, Symantec entre outras empresas de nome no segmento da tecnologia. Mas a final o que os métodos ágeis trazem de diferente? O que tem despertado tanto interesse nessas grandes empresas tecnológicas? Para entender melhor, devemos compreender o início de tudo.

No começo da década de 90, com o objetivo de simplificar os processos de desenvolvimento de software, novas abordagem chamadas de "processo leves". Tratam-se de metodologias de desenvolvimento adaptativa e flexíveis, e que são indicadas para cenários onde a mudança de requisitos é constante e os resultados precisam ser entregues ao cliente em curto espeço de tempo. A proposta destas metodologias é dividir o desenvolvimento em ciclos curtos ou interações de apenas algumas semanas, de modo que no final de cada ciclo, o cliente receba uma versão que agregue valor ao produto, como Scrum, Extreme Programming (XP), e Feature Driven Development (FDD), são alguns exemplos de metodologia ágeis que surgiram no final do século passado a melhorar a produtividade no desenvolvimento de software focando mais a entrega do que a documentação, mas não a deixando de lado.

Os desenvolvedores podem acompanhar as mudanças dos requisitos no início de cada ciclo, além e ter um acompanhamento continuo dos clientes, reduzindo assim os riscos do projeto e com isso evitando o que ao fim do projeto o cliente não tenha uma surpresa ruim para com o resultado final do seu investimento.

Devido ao grande número de referência a esses processos leves, que atingiram como resposta aos constantes fracassos de projetos utilizando abordagem tradicionais, de 11 a 13 de fevereiro de 2001, na The Longe da estação de esqui Snowbird nas montanhas Wasatch em Utah, EUA, um grupo de 17 profissionais extraordinários no desenvolvimento de software reuniu-se para esquiar, relaxar e tentar achar um denominador comum. Surgindo assim o simbólico Manifesto Ágil, uma declaração com os princípios que regem o desenvolvimento ágil

O Manifesto Ágil

Os profissionais que deram origem ao manifesto ágil foram Ken Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Kin Schwaber, Jeff Sutherland e Dave Thomas. Estamos descobrindo maneiras melhores de desenvolver softwares, fazendo-o nos mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:

- **Indivíduos e a interação entre eles** mais que processos e ferramentas;
- **Software em funcionamento** mais que documentação abrangente;
- **Colaboração com o cliente** mais que negociação contratual;
- **Responder a mudanças** mais que seguir um plano.

Mesmo havendo valor nos itens a direita, valorizamos mais os itens a esquerda. Além desses 4 valores que regem a metodologia ágil também é composto por 12 princípios:

- Nossa maior prioridade e satisfazer o cliente através da entrega continua e adiantada de software com valor agregado.
- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.
- Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência a menor escala de tempo.
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
- Construa projetos em torno de indivíduos motivados. De a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento e através de conversa face a face.
- Software funcionando e a medida primaria de progresso.
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- Continua atenção a excelência técnica e bom design aumenta a agilidade.
- Simplicidade - a arte de maximizar a quantidade de trabalho não realizado – e essencial.
- As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Métodos ágeis assumem imprevisibilidade natural do desenvolvimento de software, por isso, considera-se que o cliente também está aprendendo sobre o que precisa e, que cada feedback pode mudar de ideia e alterar o escopo do projeto. Assume-se também que estimativas de esforço e tempo de desenvolvimento são, de fato estimativa, e não devem ser tratadas como algo certo e sem margem de erro.

Pelo fato de que o desenvolvimento de software é uma ciência imprevisível, métodos ágeis se baseiam nos ciclos de feedback constante permitindo que o cliente e a equipe de desenvolvimento possam adaptar-se ás mudanças, aumentando assim as chances de sucesso do projeto.

Com intenção de promover os métodos ágeis, foi fundada a Aliança Ágil (Agile Alliance), que apoiou e realizou uma série de eventos e conferências ao redor do mundo. Com o tempo, mais e mais empresas e pessoas foram adorando métodos ágeis e atualmente milhões de pessoas consideram-se adeptas a essa metodologia.

Valores Ágeis

Conforme citado anteriormente, o Manifesto Ágil é formado por quatro valores fundamentais. O primeiro valor, que diz "**indivíduos e as interações mais entre ele que processos e ferramentas**", trata de entender que uma equipe é formada por pessoas e, que cada uma é diferente e única e possui pontos fortes e fracos, e não são apenas "recursos".

O bom relacionamento entre membros da equipe é considerado crucial, por isso, a agilidade do ambiente estimula o trabalho em equipe, a colaboração, e a comunicação constante. As equipes, geralmente, são formadas por pessoa com diferentes papéis, que se responsabilizam juntas pelo resultado do trabalho.

Processos são realizados por pessoas, e as ferramentas são utilizadas por pessoas. Se a interação entre elas não estiver bem equilibrada, provavelmente, a eficácia dos processos e ferramentas será comprometida. Por isso, nos métodos ágeis, as pessoas estão em primeiro lugar. Por outro lado, é importante ressaltar que as ferramentas também são importantes, apenso não são mais que as pessoas. Essa lógica vale para todos os valores do manifesto: o elemento da esquerda é mais importante do que o da direita, porém o da direita também é importante e relevante.

O segundo valor "**software em funcionamento mais que documentação abrangente**" é uma resposta a projetos tradicionais em que por serem realizados por fases, costumava-se passar meses produzindo apenas documentação, que por si só, não agregava muito valor, ou talvez num valor ao cliente.

A interação dos métodos ágeis permite que software em funcionamento seja entregue ao cliente em curto espaço de tempo, dessa maneira agregando valor maior em curto período de tempo. Não deixando de lado é claro a documentação pois é importante para o projeto.

O terceiro valor "**colaboração com o cliente é mais valorizada do que negociação contratual**" o objetivo da equipe ágil é entregar um produto que agregue valor; e para isso é preciso estar sempre pronto adaptar-se ás mudanças que, geralmente, ocorrem no mundo dos negócios, e consequentemente afetam uma ideia de escopo inicial que o cliente tinha do projeto a ser desenvolvido.

Contratos, geralmente, são necessários, mas muitos são protecionistas demais e procuram fechar o escopo do projeto junto com o cliente ao longo do processo de desenvolvimento, com isso acaba, por muitas vezes, gerando um produto que não atende as necessidades de quem está patrocinando o projeto. O ideal é que o contrato tenha o máximo de opções abertas para que o projeto possa mudar na medida do necessário e dessa forma o projeto seja adaptável e, realmente, gere valor ao cliente. Além do mais, é necessário a colaboração e participação do cliente durante o desenvolvimento. Métodos ágeis procuram trazer o cliente para perto da equipe, o cliente faz parte do projeto e tem um papel muito importante para que o projeto seja bem-sucedido.

O quarto e último valor do manifesto ágil, "**responder a mudanças é mais importante que seguir um plano**", diz respeito, essencialmente, à capacidade de adaptação que uma equipe ágil precisa possuir. Planejar é preciso, mas planos não precisam ser escritos em pedras, eles podem ser apagados, corrigidos, refeitos. A capacidade de adaptar-se em um mundo em constante mudança é uma qualidade essencial entregar projetos relevantes e bem-sucedidos.

Benefícios

Uma pesquisa realizada pela VersionOne.com em 2011 envolveu mais de 6.000 pessoas e organizações dos mais variados perfis na indústria de desenvolvimento de software. Ela nos mostra alguns dos principais benefícios obtidos por essas organizações após a transição para métodos ágeis. Os principais benefícios são:

- Melhor Time-to-market e Maior retorno sobre o Investimento.
- Maior satisfação do cliente e melhor gestão de mudanças de prioridades.
- Melhor visibilidade dos projetos.
- Maior produtividade.
- Equipes mais motivadas.
- Melhor disciplina na engenharia e melhor interna.
- Processo de desenvolvimento simplificado.
- Redução de riscos.
- Redução de curtos.

Scrum

Scrum é um dos métodos ágeis mais populares na atualidade e tem foco maior nos aspectos gerenciais do desenvolvimento de software. Nele, cada interação é chamada de Sprint. Geralmente cada Sprint tem um período de tempo que pode sofrer variações de poucos dias a algumas semanas. As pessoas envolvidas no processo de desenvolvimento são divididas no Scrum em três papeis principais: o Scrum Master, Product Owner (dono do produto) e a equipe.

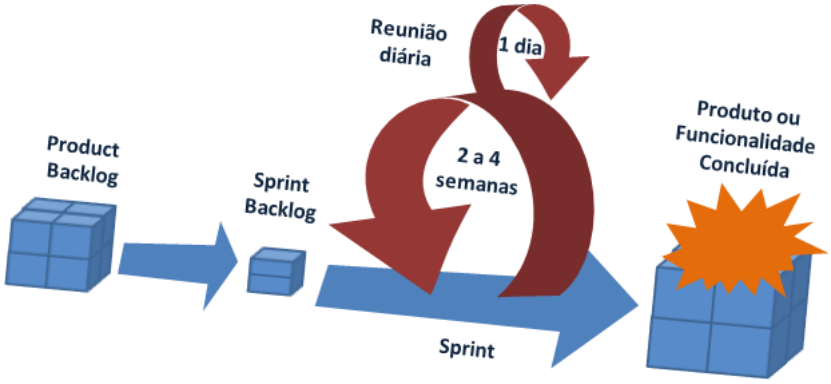

Figura 2: Scrum

O Product Owner é o maior interessado no software, é aquele que teve (ou representa quem teve) a necessidade do produto, por isso tem a visão do produto. Define o que deve ser feito, e prioriza as funcionalidades a serem desenvolvidas, mantendo-as em um artefato chamando de product backlog, que nada mais é do que uma lista de todas as funcionalidades que devem ser implementadas, ordenada por prioridade. É também responsabilidade do

Product Owner passar toda a informação de negócio que for necessária para a equipe possa transformar sua ideia em software.

A Equipe é composta por um número que varia de cinco a nove pessoas e é Cross-funcional, isto é, há pessoas dos mais variados perfis, como designer, testadores, desenvolvedores, analistas de negócio etc. integrando a equipe. O principal objetivo dela é implementar as funcionalidades que foram selecionadas para serem desenvolvidas na intenção de entrega-las em funcionamento ao final desse período.

Por fim o Scrum Master, também chamado de facilitados, é responsável por manter o processo em funcionamento, assegurando que todas as regras estão sendo aplicadas, e remover os impedimentos da equipe, isto é resolver qualquer problema que possa atrapalhar o processo de desenvolvimento, garantindo assim que o objetivo da interação seja atingindo. É importante ressaltar que o Scrum Master não atua como gerente ou chefe da equipe porque ela é auto organizável, não determinando o que cada membro da equipe deve ou não fazer: a equipe se compromete com a entrega das funcionalidades e, então, se auto organiza, definindo por quem e qual momentos as tarefas serão realizadas.

Toda Sprint começa com uma reunião de planejamento, que é dividida em duas partes. A primeira delas tem um enfoque mais estratégico, na qual se decide o que será feito, quais funcionalidades serão implementadas e define-se uma meta para a Sprint. O Product Owner apresenta os itens do produck backlog, e a equipe obtém informações suficientes sobre cada um dos itens, dessa forma, é possível fornecer uma estimativa que demonstre um tamanho ou número de horada para o trabalho e assim seja definido quantas e quais tarefas poderão ser desenvolvidas no Sprint.

Logo após a definição das tarefas a serem desenvolvidas no Sprint, a equipe utiliza a segunda parte da reunião, que tem um enfoque mais tático, para definir como serão feitas as tarefas. Portanto é feita uma análise de tarefa por tarefa com mais detalhes, mais informações de negócio são apresentadas e é possível tomar diversas decisões de negócio e decisões técnicas.

Ao finalizar a reunião de planejamento a equipe começa a trabalhar nas tarefas, respeitando a prioridade: fazendo sempre primeiro as tarefas mais importantes, que representa maios valor para o produto de acordo com o Product Owner. Todos os dias é feita uma reunião para que a equipe possa conversar sobre como está o andamento do Sprint.

Ao decorrer da Sprint a equipe mantém sempre em mente sua meta e quando o andamento foge do que foi previsto, a equipe pode negociar o escopo com o Product Owner, de forma que não atrapalhe a meta.

Ao fim da Sprint, mais duas reuniões acontecem: a reunião de revisão e a reunião de retrospectiva. Na primeira a equipe apresenta ao Porduct Owner o trabalho que foi desenvolvido durante o Sprint, para que ele possa oferecer feedback, e aprovar ou não tudo que foi produzido. Já na segunda são apresentados e discutidos os principais acontecimentos da Sprint e com isso é apresentado as principais lições aprendidas e melhorias que podem ser aplicadas ao processo.

Extreme Programming

O método ágil Extreme Programming, que em português significa Programação Extrema, mais é conhecido apenas como XP, foi cirando por Kent Beck nos anos 90, e é um dos métodos ágeis que melhor cobre aspectos técnicos de desenvolvimento de software como codificação, design e teste.

De acordo com o XP, no decorrer do processo de desenvolvimento há quatro atividades triviais a serem feitas: Ouvir, Desenhar, Codificar e Testar. É preciso ouvir porque desenvolvedores nem sempre possuem o conhecimento desejável do negócio para que se possa criar o software. O Planning Game é uma reunião que acontece uma vez a cada interação, em que o principal objetivo é decidir quais funcionalidades serão implementadas na interação e aprender mais sobre elas.

O cliente escreve **histórias de usuário** em cartões que representam a necessidade de funcionalidade a serem desenvolvidas, e explica para os desenvolvedores tudo o que for preciso para que eles possam implementa-la. Um bom design é uma ótima ferramenta para que todos possam compreender melhor os sistemas complexos. O principal objetivo é manter o design sempre simples, evitando aumentar a complexidade com a criação de funcionalidades que não sejam realmente necessárias.

Ao manter o design simples e fazer somente aquilo que for necessário, é poupado tempo, pois é deixado de escrever código que não é preciso e com isso terá mais tempo para se concentrar na qualidade do que realmente importa e que vai agregar valor para o produto final.

Codificar é uma atividade essencial, afinal de contas, sem código não existe software algum. Nessa etapa, é extremamente importante manter contato com o cliente para que possa ser feito o feedback e responder as dúvidas que surgirem durante o desenvolvimento.

Os testes de unidades são escritos antes do código de produção, todo código fonte que será executado em produção é desenvolvidos em pares, que consiste basicamente em codificar em dupla, durante a implementação, um programador age como piloto (digitando o código) e o outro age como copiloto (revisando o que está sendo digitado, apontando problemas e pensando na solução como um todo). A cada determinado ciclo de tempo, os profissionais invertem os papéis. Dessa forma o piloto passa a ocupar o papel de copiloto e vice-versa.

A integração do código fonte é realizada frequentemente através de prática de integração continua que consistem em algo simples: o desenvolvedor integra o código alterado e/ou desenvolvido ao projeto principal na mesma frequência com que as funcionalidades são desenvolvidas, sendo feito muitas vezes ao dia ao invés de apenas uma vez. O objetivo principal da integração continua é verificar se as alterações ou novas funcionalidades não criaram novos defeitos no projeto já existente. A prática da integração contínua pode ser feita através de processos manuais ou automatizados, utilizando ferramentas.

E por fim o código fonte é coletivo, ou seja, pertence a todos os membros da equipe, e deve ser escrito de acordo com os padrões definidos pelo próprio time. Testar é uma atividade de extrema importância para garantir a qualidade do software, todos os códigos devem possuir teste de unidade, e todos os teste devem ser executados com sucesso antes que uma entrega seja feita. Devido ao foco técnico do XP e ao foco gerencial do Scrum, muitas empresas combinam os dois para obter um processo mais completo, visando um processo mais eficiente.

O Kanban

Kanban é um método ágil, diferente da maioria dos métodos ágeis, não possui interações. Ao invés disso, desacopla o planejamento, priorização, desenvolvimento e entrega, de forma que cada uma dessas atividade possa ter sua própria cadência para melhor se ajustar à realidade e necessário que o processo demanda.

Cadências são repetições que se sucedem em intervalos regulares. Este é um conceito do método Kanban que determina o ritmo de um determinado tipo de evento. Priorização, entregas, retrospectiva e qualquer evento recorrente podem ter sua própria cadência.

Kanban é um processo contínuo e não é preciso que todo o trabalho esteja concluído para que uma entrega seja realizada. No momento da entrega, algumas tarefas estarão prontas e outras em processo; as que estiverem prontas farão parte da entrega, e as que não estiverem ficaram para a próxima.

O Kanban está bastante relacionado com o conceito de Pull Systems (sistemas de produção puxados, esse sistema exige que cada operação do processo seja alimentada pela demanda da etapa anterior, estabelecendo uma ligação direta entre o consumo real do cliente e da quantidade produzida. Dessa forma, o fato de um item ter sido vendido, geraria demandas para a fábrica de outro. Em uma abordagem de desenvolvimento de software, poderíamos dizer que a demanda para se trabalhar em uma nova funcionalidade seria gerada quando alguma outra fosse entregue.

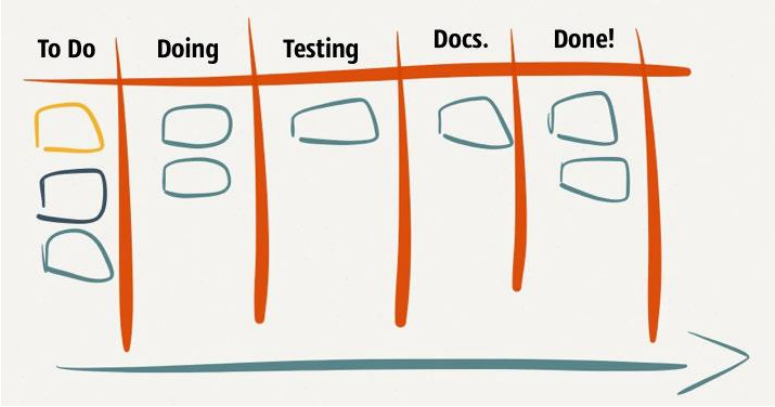

Figura 3: Kanban

Kanban é uma palavra japonesa que significa cartão sinalizador. É utilizada no Sistema Toyota de produção e também por diversas empresas que aderiram à filosofia Lean para representar, em um sistema puxado de produção, um sinal para que se puxe mais trabalho, ou seja, para que o processo seja alimentado.

Sendo este um método de desenvolvimento de software menos prescritivo, que possui apenas três prescrições: Visualizar o fluxo de trabalho (workflow), limitar o trabalho em progresso e gerenciar e medir o fluxo.

Kanban incentiva a adaptação baseada em contexto. Deste modo, parte-se da premissa que não existe uma única solução que resolva todos os problemas de todos os diferentes contextos, e por isso há a necessidade de adaptação, em que cada processo deve se adequar as necessidades do contexto a que pertence.

Kanban também recomenda que se meça o **lead time**, tempo médio para se completar um item, de forma que o processo deve ser continuamente otimizado para tornar esse tempo cada vez menor e mais previsível.

Conclusão

Por todos esses aspectos surgi a grande dúvida de qual é o melhor método a ser seguido, Scrum, XP ou Kanban? Com tudo, esses métodos são ferramentas para processos, a definição de ferramenta é: qualquer coisa que possa era usada para atingir uma tarefa ou objetivo. Com isso não há uma maneira de escolher o melhor método, pois a melhor escolha depende do contesto e no meio que o projeto está sendo implementado.

Nenhum método é completo e tal pouco perfeito, por isso, é fortemente aplicado a combinação de dessas ferramentas e com isso tirando o melhor aproveitamento o que funciona melhor para o contexto de cada projeto, ou seja, considerando a cultura organizacional, as habilidades da equipe e as restrições do projeto. É comum, por exemplo, que se adote práticas de engenharias de XP, como programação em par e integração contínua, em times que fazem o uso do Scrum.

Todo projeto de desenvolvimento de software é diferente, possui pessoas com habilidade diferentes, níveis de experiência diferentes, orçamentos, prazos, escopo e riscos diferentes. Os valores da organização e suas metas também são específicos, e com toda essa variação, as restrições também são diferentes. Portanto não há um manual a ser seguido pois cada caso é um caso.

Referência

TAVARES. Breno Gontijo, DA SILVA. Carlos Eduardo Sanches. Análise bibliometrica de artigos científicos sobre a utilização de metodologias ágeis na gestão de projetos. 2012

O manifesto ágil: http://www.manifestoagil.com.br/

ALMEIDA. Vinicius. Introdução ao desenvolvimento ágil. http://www.devmedia.com.br/introducao-ao-desenvolvimento-agil/5916

DE CARVALHO. Bernardo Vasconcelos, MELLO Carlos Henrique pereira. Revisão, análise e classificação da literatura sobre o método de desenvolvimento de produtos ágil Scrum. 2009

GOMES. André Faria. Agile Desenvolvimento de software com entregas frequentes e foco de negócio. Ed Casa do Código

BONFIM Márcio. Pair Programming: Vantagens e desvantagens da programação em par. http://www.devmedia.com.br/pair-programming-vantagens-e-desvantagens-da-programacao-em-par/30537