UNIVERSIDADE DO ESTADO DE MINAS GERAIS-UEMG
NÚCLEO ACADÊMICO DE TECNOLOGIA E ENGENHARIA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Métodos Ágeis

SCRUM

Alunos
Anderson Veloso dos Santos
Junior César da Silva
Maria Andressa de Paula Silva
Rafael de Oliveira Romano

PASSOS/MG
2016

1 Métodos Ágeis

Entre os anos de 1980 e 1990, existia uma grande preocupação com o desenvolvimento de *software* pela comunidade de engenharia de *software*. Isso porque a ideia era de que para um *software* ser desenvolvido, necessitava de um projeto seguido à risca para se chegar em um produto final de qualidade e de acordo com os requisitos do cliente. Porém, esta visão era para sistemas grandes, ou seja, sistemas críticos; que eram desenvolvidos por uma equipe com grande número de desenvolvedores.

Com isso, o tempo gasto planejando o *software* era maior que o tempo gasto desenvolvendo e realizando testes. Mudanças na definição de requisitos, faziam com que gerasse retrabalho e isso afetava todo o projeto.

Com base neste tipo de desenvolvimento, alguns desenvolvedores de *software* na década de 1990, chegaram à conclusão que a equipe de desenvolvimento devia se concentrar somente no *software* e não mais em seu projeto e documentação. Foi então, que surgiu o conceito de desenvolvimento de *software* baseado em Métodos Ágeis.

Para a especificação, desenvolvimento e entrega de *software*, os métodos ágeis contam com a abordagem iterativa. Foi criada para auxiliar em desenvolvimento de aplicações em que os requisitos mudam durante o processo. A ideia é apoiar aplicações de negócios, onde a entrega do produto ao cliente é feita rapidamente, e assim podendo incluir novos requisito e/ou alterações.

O método ágil mais conhecido é o *Extreme Programming* (XP), entre outros como *Scrum*. Todos os métodos, apesar de se basearem em uma entrega incremental, compartilhar os mesmos princípios, tais como:

- Envolvimento do Cliente: estão sempre envolvidos no processo de desenvolvimento, fornecendo os requisitos e avaliando as iterações do sistema.
- Entrega incremental: o cliente especifica os requisitos a cada novo incremento do sistema.
- Pessoas: reconhecer e explorar as habilidades da equipe.
- Mudanças: projetar o sistema para acomodar mudanças, tendo em mente que os requisitos sempre irão mudar.
- Simplicidade: eliminar a complexidade do sistema, ele deve ser o mais simples possível.

Segundo DeMarco e Boehm (2002), uma abordagem híbrida entre métodos ágeis e desenvolvimento baseado em planos pode ser uma melhor solução.

Os princípios básicos dos métodos ágeis se tornam difíceis de entender na prática pelos seguintes motivos:

- Cliente: o sucesso depende do cliente disposto a representar os *stakeholders* e dedicar tempo no processo de desenvolvimento.
- Equipe: membros podem não ter a personalidade necessária para lidar com esse tipo de envolvimento intenso dos métodos ágeis ou ainda não interagir bem com outros membros da equipe.
- Mudanças: dificuldade na priorização de mudanças pelo fato de cada *stakeholders* ter prioridade diferente em cada mudança.
- Simplicidade: por pressão dos cronogramas, pode não ser possível atender a simplificações desejadas no sistema.

O contrato entre desenvolvedores e cliente pode ser feito com base em pagamentos pelo tempo de desenvolvimento do sistema, em vez do desenvolvimento de um conjunto de requisitos. Além disso, os métodos ágeis são adequados para desenvolvimentos de sistemas de pequenas e médias empresas; não podendo ser usados para sistemas críticos.

2 Diferenças entres os Métodos Ágeis e os Modelos Clássicos de Desenvolvimento de Software

Desenvolver e fazer uma entrega rápida de um *software* muitas vezes se torna o requisito mais crítico. Em um ambiente com mudanças constantes, os clientes acham impossível prever várias necessidades. Muitos requisitos só se tornam claros após a entrega do produto final aos usuários.

Um modelo clássico de desenvolvimento de *software* (requisitos, projeto, construção e teste) não está voltado para o desenvolvimento rápido de *software*. A medida que os requisitos mudam, gera um retrabalho. Um modelo como por exemplo o cascata, é geralmente prolongado e só entregue ao cliente muito depois de ter sido especificado.

Processo de desenvolvimento rápido são projetados para desenvolver *software* útil rapidamente. São processos iterativos onde especificação, projeto, desenvolvimento e teste são intercalados. Assim, o *software* é entregue em partes, em incrementos com funcionalidades do sistema. Suas características são:

- O documento de requisitos define características mais importantes. Os processos de especificação, projeto e implementação são concorrentes e a documentação é minimizada ou gerada automaticamente.
- Sistema desenvolvido em incrementos.
- Interface criada rapidamente por desenhos e inserções de ícones.

3 Definição do Scrum

Scrum: Um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. *Scrum* é: leve, simples de entender e extremamente difícil de dominar.

Scrum é um *framework* estrutural que está sendo usado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990. *Scrum* não é um processo ou uma técnica para construir produtos; em vez disso, é um *framework* dentro do qual você pode empregar vários processos ou técnicas. O *Scrum* deixa clara a eficácia relativa das práticas de gerenciamento e desenvolvimento de produtos, de modo que você possa melhorá-las.

O *framework Scrum* consiste nos *times* do *Scrum* associadas a papéis, eventos, artefatos e regras. Cada componente dentro do *framework* serve a um propósito específico e é essencial para o uso e sucesso do *Scrum*.

3.1 Teoria do Scrum

O empirismo afirma que o conhecimento vem da experiência e de tomada de decisões baseadas no que é conhecido. O *Scrum* emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos. Três pilares apoiam a implementação de controle de processo empírico: transparência, inspeção e adaptação.

A transparência está para aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados. Esta transparência requer aspectos definidos por um padrão comum para que os observadores compartilharem um mesmo entendimento do que está sendo visto.

A inspeção mostra que os usuários *Scrum* devem, frequentemente, inspecionar os artefatos Scrum e o progresso em direção a detectar variações. Esta inspeção não deve, no entanto, ser tão frequente que atrapalhe a própria execução das tarefas. As inspeções são mais benéficas quando realizadas de forma diligente por inspetores especializados no trabalho a se verificar.

Já a adaptação ressalta que se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, e que o produto resultado será inaceitável, o processo ou o material sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve possível para minimizar mais desvios.

3.2 O Time Scrum

O *Time Scrum* é composto pelo *Product Owner*, o Time de Desenvolvimento e o *Scrum Master*. Times Scrum são auto organizáveis e multifuncionais. Times auto

organizáveis escolhem qual a melhor forma para completarem seu trabalho, em vez de serem dirigidos por outros de fora do Time. Times multifuncionais possuem todas as competências necessárias para completar o trabalho sem depender de outros que não fazem parte da equipe. O modelo de time no *Scrum* é projetado para aperfeiçoar a flexibilidade, criatividade e produtividade.

3.2.1 O Product Owner

O *Product Owner*, ou dono do produto, é o responsável por maximizar o valor do produto e do trabalho do Time de Desenvolvimento. O *Product Owner* é a única pessoa responsável por gerenciar o *Backlog* do Produto.

3.2.2 O Time de Desenvolvimento

O Time de Desenvolvimento consiste de profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto “Pronto” ao final de cada *Sprint*. Somente integrantes do Time de Desenvolvimento criam incrementos.

Os Times de Desenvolvimento são estruturados e autorizados pela organização para organizar e gerenciar seu próprio trabalho. A sinergia resultante aperfeiçoa a eficiência e a eficácia do Time de Desenvolvimento como um todo.

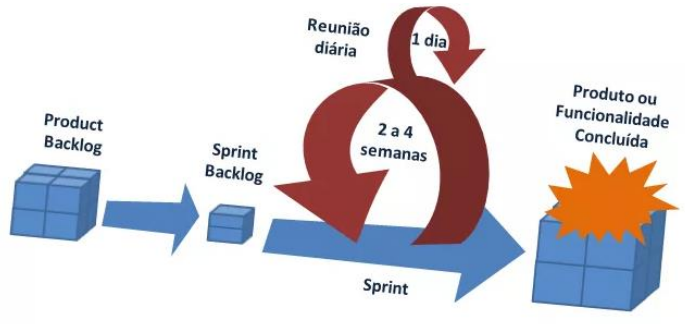
3.2.3 O Scrum Master

O *Scrum Master* é responsável por garantir que o Scrum seja entendido e aplicado. O Scrum Master faz isso para garantir que o Time Scrum adere à teoria, práticas e regras do Scrum. O Scrum *Master* é um servo-líder para o Time Scrum. O Scrum Master ajuda aqueles que estão fora do Time Scrum a entender quais as suas interações com o Time Scrum são úteis e quais não são. O Scrum Master ajuda todos a mudarem estas interações para maximizar o valor criado pelo Time Scrum.

Normalmente o Scrum Master não tem autoridade para exercer o controle sobre a equipe, de modo que este papel não é o mesmo que o papel tradicional do Gerente de Projeto ou Gerente de Desenvolvimento. O Scrum Master age como um líder, não como um gerente.

3.3 Eventos Scrum

Eventos prescritos são usados no Scrum para criar uma rotina e minimizar a necessidade de reuniões não definidas no Scrum. Todos os eventos são eventos *time-boxed*, de tal modo que todo evento tem uma duração máxima. Uma vez que a Sprint começa, sua duração é fixada e não pode ser reduzida ou aumentada. Os eventos restantes podem terminar sempre que o propósito do evento é alcançado, garantindo que uma quantidade adequada de tempo seja gasta sem permitir perdas no processo.



3.3.1 Sprint

O coração do *Scrum* é a *Sprint*, um time-boxed de um mês ou menos, durante o qual um “Pronto”, versão incremental potencialmente utilizável do produto, é criado. Sprints tem durações coerentes em todo o esforço de desenvolvimento. Uma nova Sprint inicia imediatamente após a conclusão da *Sprint* anterior.

As *Sprints* são compostas por uma reunião de planejamento da Sprint, reuniões diárias, o trabalho de desenvolvimento, uma revisão da Sprint e a retrospectiva da Sprint. Cada Sprint pode ser considerada um projeto com horizonte não maior que um mês. Como os projetos, as Sprints são utilizadas para realizar algo. Cada Sprint tem a definição do que é para ser construído, um plano projetado e flexível que irá guiar a construção, o trabalho e o resultado do produto.

3.3.2 Reunião de Planejamento da Sprint

Reunião de planejamento da Sprint possui um time-box com no máximo oito horas para uma Sprint de um mês de duração. Para Sprints menores, este evento é usualmente menor. O Scrum Master garante que o evento ocorra e que os participantes entendam seu propósito. O Scrum Master ensina o Time Scrum a manter-se dentro dos limites do time-box. A reunião de planejamento da Sprint responde as seguintes questões:

- O que pode ser entregue como resultado do incremento da próxima Sprint?
- Como o trabalho necessário para entregar o incremento será realizado?

3.3.3 Reunião Diária

A Reunião Diária do Scrum é um evento time-boxed de 15 minutos, para que o Time de Desenvolvimento possa sincronizar as atividades e criar um plano para as próximas 24 horas. Esta reunião é feita para inspecionar o trabalho desde a última Reunião Diária, e prever o trabalho que deverá ser feito antes da próxima Reunião Diária. Durante a reunião os membros do Time de Desenvolvimento esclarecem:

- O que eu fiz ontem que ajudou o Time de Desenvolvimento a atender a meta da Sprint?
- O que eu farei hoje para ajudar o Time de Desenvolvimento atender a meta da Sprint?
- Eu vejo algum obstáculo que impeça a mim ou o Time de Desenvolvimento no atendimento da meta da Sprint?

3.3.4 Revisão da Sprint

A Revisão da Sprint é executada no final da Sprint para inspecionar o incremento e adaptar o Backlog do Produto se necessário. Durante a reunião de Revisão da Sprint o Time Scrum e as partes interessadas colaboram sobre o que foi feito na Sprint. Com base nisso e em qualquer mudança no Backlog do Produto durante a Sprint, os participantes colaboram nas próximas coisas que podem ser feitas para aperfeiçoar valor. O resultado da Reunião de Revisão da Sprint é um Backlog do Produto revisado que define o provável Backlog do Produto para a próxima Sprint. O Backlog do Produto pode também ser ajustado completamente para atender novas oportunidades.

3.3.5 Retrospectiva da Sprint

A Retrospectiva da Sprint é uma oportunidade para o Time Scrum inspecionar a si próprio e criar um plano para melhorias a serem aplicadas na próxima Sprint.

A Retrospectiva da Sprint ocorre depois da Revisão da Sprint e antes da reunião de planejamento da próxima Sprint. O Scrum Master garante que o evento ocorra e que os participantes entendam seu propósito, ensina todos a mantê-lo dentro do time-box. O Scrum Master participa da reunião como um membro auxiliar do time devido a sua responsabilidade pelo processo Scrum. O propósito da Retrospectiva da Sprint é: inspecionar como a última Sprint foi em relação às pessoas, aos relacionamentos, aos processos e às ferramentas; identificar e ordenar os principais itens que foram bem e as potenciais melhorias; e criar um plano para implementar melhorias no modo que o Time Scrum faz seu trabalho.

Ao final da Retrospectiva da Sprint, o Time Scrum deverá ter identificado melhorias que serão implementadas na próxima Sprint.

3.3.6 Artefatos do Scrum

Os artefatos do Scrum representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação. Os artefatos definidos para o Scrum são especificamente projetados para maximizar a transparência das informações chave de modo que todos tenham o mesmo entendimento dos artefatos.

3.4 Backlog do Produto

O Backlog do Produto é uma lista ordenada de tudo que deve ser necessário no produto, e é uma origem única dos requisitos para qualquer mudança a ser feita no produto. O Product Owner é responsável pelo Backlog do Produto, incluindo seu conteúdo, disponibilidade e ordenação. Em seguida, ele garante que os itens do Backlog são colocados na sequência correta (usando fatores como valor, custo, conhecimento e risco), de modo que os itens de alto valor, aparecerá no topo do backlog do produto e os itens de menor valor aparecer em direção ao fundo.

O Backlog do Produto lista todas as características, funções, requisitos, melhorias e correções que formam as mudanças que devem ser feitas no produto nas futuras versões. Os itens do Backlog do Produto possuem os atributos de descrição, ordem, estimativa e valor.

3.4.1 Backlog da Sprint

O Backlog da Sprint é um conjunto de itens do Backlog do Produto selecionados para a Sprint, juntamente com o plano para entregar o incremento do produto e atingir o objetivo da Sprint. O Backlog da Sprint é a previsão do Time de Desenvolvimento sobre qual funcionalidade estará no próximo incremento e sobre o trabalho necessário para entregar essa funcionalidade em um incremento “Pronto”.

O Backlog da Sprint é um plano com detalhes suficientes que as mudanças no progresso sejam entendidas durante a Reunião Diária. O Time de Desenvolvimento modifica o Backlog da Sprint ao longo de toda a Sprint, e o Backlog da Sprint vai surgindo durante a Sprint.

3.4.2 Definição de “Pronto”

Quando o item do Backlog do Produto ou um incremento é descrito como “Pronto”, todos devem entender o que o “Pronto” significa. Embora, isso varie significativamente de um extremo ao outro para cada Time Scrum, os integrantes devem ter um entendimento compartilhado do que significa o trabalho estar completo, assegurando a transparência. Esta é a “Definição de Pronto” para o Time Scrum e é usado para assegurar quando o trabalho está completado no incremento do produto.

Se a definição de “pronto” para um incremento é parte das convenções, padrões ou diretrizes de desenvolvimento da organização, todos os Times Scrum devem segui-la como um mínimo. Se “pronto” para um incremento não é uma convenção de desenvolvimento da organização, o Time de Desenvolvimento do Time Scrum deve definir uma definição de “pronto” apropriada para o produto. Se há múltiplos Times

Scrum trabalhando no lançamento do sistema ou produto, os times de desenvolvimento de todos os Times Scrum devem mutuamente definir a definição de “Pronto”.

Cada incremento é adicionado a todos os incrementos anteriores e completamente testado, garantindo que todos os incrementos funcionam juntos. Com um Time Scrum maduro, é esperado que a sua definição de “Pronto” seja expandida para incluir critérios mais rigorosos de alta qualidade.

4 Vantagens no uso de Scrum

- Motivação: os programadores se sentem muito mais motivados devido ao seu interesse de entregar o Sprint no prazo.

- O projeto pode ser visualizado: dentro da organização o projeto pode ser observado por todos. Em outras metodologias esta possibilidade não existia.

- Ausência significante de bugs: como a qualidade é mais importante do que o prazo de entrega, o produto apresenta uma diminuição significativa de erros (bug).

- Alterar as prioridades: os programadores podem manejar as prioridades sem problemas, garantindo assim que Sprints que ainda não foram finalizados possam ser alterados sem problemas.

4.1 Desvantagens no uso de Scrum

- Prazo: como a qualidade é mais importante do que o resultado, pode ser que os prazos não sejam estipulados de forma coerente, levando a um atraso do resultado final, o que pode deixar os clientes com uma certa raiva, mas isso pode ser ajustado em equipe.

- Desordem nas funções: a presença de papéis indefinidos nas funções presentes no projeto pode dar alguns problemas relacionados a comunicação interna e deixar os programadores confusos quanto as suas tarefas.

- Ausência de documentação: a falta de documentações sobre o andamento do projeto pode ser um grande problema. Por isso é importante documentar aspectos que sejam verdadeiramente importantes, mas não deixar de lado a documentação de tudo o que está acontecendo. Posteriormente, pode ficar difícil voltar em um determinado instante do projeto e lidar com a situação de não ter aquele momento documentado.

5 Quando utilizar Scrum no desenvolvimento de software

- Indivíduos e relacionamentos ao invés de processos e ferramentas, as chances de sucesso em um projeto são muito grandes maiores em times onde há uma boa comunicação, onde há entendimento e cooperação mutua.

- Software funcionando ao invés de extensa documentação, com uma versão funcional do produto o cliente pode interagir e avaliar o software. O gerente de produto

pode usar o software para avaliar se suas expectativas foram cumpridas e sua forma de comunicação com o time foi eficiente.

- A negociação com o cliente ao invés de negociação de contrato, em projeto Scrum, o cliente e parte integra, do time trabalhando diariamente com os seus desenvolvedores e Scrum máster. No projeto ideal, o Scrum máster e desenvolvedores não precisam de tomar uma decisão sobre a regra de negócios o cliente está sempre disponível.

- Reação a mudanças ao invés de seguir um plano, ao invés do planejamento, o importante que o time possa sofrer mudanças que tornam o produto final melhor.

5.1 Quando não utilizar Scrum no desenvolvimento de software

- Quando o gerente nomear um líder de equipe para ser Scrum master, Scrum máster pode ainda vestir a camisa do gerente expressando conduta mandatória sobre o time o que não é recomendado, visto que o time tem que saber gerenciar.

- O product owner não é presente ou então trabalha em muitos times ao mesmo tempo, o time com certeza terá duvidas no decorrer da Sprint, o time terá sugestões e a probabilidade de fazer algo que o cliente não queira é muito alta.

6 Exemplo prático de Scrum

Para exemplificar o processo, utilizaremos um exemplo: o desenvolvimento de um sistema web de streaming de vídeo para a Internet. Os papéis são:

- Proprietário do Produto: Rafael.
- Scrum Master: Anderson.
- Equipe de Desenvolvimento: Andressa e Júnior.

6.1 Escopo do Projeto

Definidos os papéis da equipe, o Proprietário do Produto Rafael, depois de algumas visitas ao cliente e pesquisas de requisitos, define o escopo do projeto:

O software será um sistema Web de streaming de vídeo, que permitirá usuários na Internet mandem seus vídeos, que serão armazenados no sistema e poderão ser gerenciados por seus donos e vistos pelo resto do mundo através das visitas ao site. O sistema terá como funcionalidades principais a conversão dos vídeos mandados para um codec leve que permite qualidade e rapidez na visualização pela Internet.

6.2 Product Backlog

Com as funcionalidades levantadas, o Proprietário do Produto Rafael monta o Product Backlog, com as prioridades definidas de acordo com o valor de mercado/importância para o cliente.

6.3 Reunião de Planejamento do Sprint

Com o Product Backlog definido, a reunião de planejamento é feita, o Proprietário do Produto Rafael apresenta o projeto aos demais membros da equipe SCRUM e toda a equipe define a quantidade de horas que cada tarefa deverá ocupar.

6.4 Início do Sprint

Com as metas preparadas e as tarefas bem definidas, chega a hora de começar o ciclo de desenvolvimento, o Sprint. O objetivo do primeiro Sprint será apresentar uma interface básica onde os usuários poderão se cadastrar, mandar e visualizar vídeos em uma interface crua. Consideramos o esqueleto do sistema.

6.5 Reuniões diárias SCRUM

O Scrum Master Anderson irá acompanhar o desenvolvimento através de reuniões diárias para se certificar que os desenvolvedores Andressa e Junior estejam completando suas tarefas, estejam bem de saúde, bem comprometidos com o projeto e se esforçando.

6.6 Burndown Chart

Com as reuniões diárias, o Scrum Master Anderson poderá alimentar o Burndown Chart. Com o Burndown Chart, podemos ver o andamento do projeto ao longo do seu ciclo de desenvolvimento (Sprint). Também, no meio do projeto, podemos calcular facilmente a velocidade com que o projeto está andando e assim estimar uma data para que o Sprint seja concluído.

6.7 Revisão final do Sprint

Ao final do ciclo de desenvolvimento (Sprint), toda a equipe se reúne para ver quais são os resultados. O Proprietário do Produto Rafael identifica todo o progresso e revisa o programa. Ele, junto com o cliente, concorda que os itens especificados para o Sprint foram completos e esta primeira versão do sistema web é satisfatória.

Referências

Ágil e Alliance. Disponível em:<http://www.agilealliance.org/article/articles_by_category/17 >. Acessado em 01 de Junho. 2016.

Code Project. Disponível em:<http://www.codeproject.com/KB/architecture/scrum.aspx>. Acessado em 01 de Junho. 2016.

Scrum Alliance. Disponível em: < http://www.scrumalliance.org/>. Acessado em 01 de Junho. 2016.

Scrum Guides. Disponível em: < http://www.scrumguides.org/>. Acessado em 25 de Maio. 2016.

SOMMERVILLE, Ian. **Engenharia de Software**. 8ª edição. São Paulo, Pearson, 2008.

VIEIRA, Denisson. **Scrum: A Metodologia Ágil Explicada de forma Definitiva.** Disponível em: < http://www.mindmaster.com.br/scrum/>. Acessado em 01 de Junho. 2016.