Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

METODOLOGIA ÁGIL NA GESTÃO DE PROJETOS DE SOFTWARE

Maurício Badoco[1]

Renato Inácio de Almeida[2]

Carlos Alberto de Lucas[3]

Resumo

O gerenciamento de um projeto de software é a habilidade de conseguir controlar todas as variáveis e parâmetros que envolvem o ambiente de desenvolvimento. O sucesso desse gerenciamento está diretamente ligado à metodologia que está sendo empregada, bem como a sua aceitação pelas equipes de desenvolvimento. Neste contexto, o objetivo deste artigo é apresentar um estudo sobre as metodologias ágeis, com enfoque nas duas principais metodologias: Scrum e Extreme Programming. Desta forma, este artigo foi elaborado com base no levantamento bibliográfico, sendo estruturado da seguinte forma: apresentação do conceito de metodologia ágil, passando para o entendimento de Scrum, em seguida para o XP, finalizando com a comparação entre a metodologia tradicional e a metodologia clássica. Demonstra que é crescente a utilização das metodologias ágeis entre as empresas desenvolvedoras de software. Concluindo que as metodologias ágeis estão num processo de evolução e constatação da sua eficiência quanto aos seus resultados.

Palavras-chave: *Extreme programming*. Gerenciamento de projetos de software. Metodologias de desenvolvimento de software. Scrum.

Abstract

The management of a software project is the ability to control all the variables and parameters that involves the environment of development. The success of this management is directly linked to the methodology being used, as well as its acceptance by the development teams. In this context, the purpose of this article is to present a study on agile methodologies, focusing on the two main methodologies: Scrum and Extreme Programming. Thus, this article was written based on the bibliographical survey, being structured as it follows: presentation of the concept of agile methodology, passing to the understanding of Scrum, then to XP, ending with the comparison between the traditional methodology and the classical methodology. It demonstrates that the use of agile methodologies among software development companies is increasing. Concluding that the agile methodologies are in a process of evolution and verification of their efficiency as well as their results.

Keywords*: Extreme programming. Scrum. Software development methodologies. Software Project Management.*

[1] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: mauriciobadoco@hotmail.com

[2] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: renatoadsfatecfranca@gmail.com

[3] Docente do curso de Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: projetos@profcarloslucas.com.br

1 Introdução

As empresas desenvolvedoras de sistemas buscam novas tecnologias para que possam ter um diferencial de competitividade no mercado. Assim, o gerenciamento de projetos tornou-se essencial, fazendo com que novos métodos fossem aplicados para garantir qualidade e os prazos estabelecidos.

Como os projetos estão em constantes mudanças durante a sua execução, as técnicas de desenvolvimento ágil tornaram-se fundamentais para auxiliar as equipes com os desafios encontrados. Além de ter como objetivo proporcionar a satisfação do cliente, com a produção de software com qualidade e o cumprimento de prazos.

Segundo Cohn (2011), muitas empresas estão tentando tornar suas equipes de desenvolvimento mais ágeis, pois as mesmas vêm produzindo software com mais qualidade e que atendem melhor os requisitos impostos pelos clientes. Equipes ágeis tendem a introduzir seus produtos no mercado com muito mais rapidez, levando em consideração principalmente a satisfação dos clientes com os requisitos e objetivos do produto.

2 Metodologias ágeis

A Engenharia de Software está em constante evolução, sempre acompanhando as tendências atuais da tecnologia. Os modelos tradicionais que foram criados e implementados há tempos atrás, estabelecem documentações excessivamente detalhadas e que demandam muito tempo para seu preenchimento, além do tempo na execução do projeto de software.

Outro ponto importante nos modelos tradicionais, está relacionado a necessidade de uma equipe com membros em quantidade considerável para cumprir prazos, além de se observar que no processo de desenvolvimento há sempre alterações de projeto.

Sommerville (2004) define “processo de software” como um conjunto de atividades cujo objetivo é o desenvolvimento ou a evolução de um software. Elas necessitam estar alinhadas e as pessoas que fazem parte do projeto estarem engajadas e sabendo qual é a sua função no processo para garantir que no final dará tudo certo. As metodologias definem os passos para que o processo de desenvolvimento atinja o padrão de qualidade pretendida.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

Neste contexto, as metodologias ágeis surgiram com a finalidade de se desenvolver *software* de forma mais direta e menos burocrática. Em 2001 surgiu o Manifesto Ágil cujo os objetivos são justamente atender as inovações contínuas, adaptabilidade de produtos, entregas com cronograma reduzido, adaptabilidade do processo e das pessoas e por fim resultados confiáveis (Martins, 2007).

Em fevereiro de 2001, nos Estados Unidos, um grupo de dezessete pessoas especialistas em processo de desenvolvimento de software, se reuniram com a finalidade de encontrarem uma forma mais enxuta e descomplicada para o processo de desenvolvimento de software. Neste encontro foi proposto para que cada um explicasse a forma como era desenvolvido os trabalhos. A partir de então foi concebida a Aliança Ágil, sendo firmada o que ficou conhecido como Manifesto Ágil que tem como base os seguintes termos (Agile Alliance, 2001):

- Indivíduos e interações são mais importantes do que processos e ferramentas.
- Software executando é mais importante que uma documentação extensa.
- O relacionamento com o cliente é mais importante que a negociação do contrato.
- Responder às mudanças rapidamente é mais importante que seguir o planejamento.

Além da aceitação de mudanças, as metodologias ágeis propõem outros princípios que trazem mais dinamismo para o processo de desenvolvimento, incluindo a proximidade com o cliente, o que prioriza sua satisfação, e a motivação da equipe, para que desempenhem seus papéis da melhor forma possível.

Desta forma, para que o desenvolvimento ocorra não há a necessidade da realização de um plano completo daquilo que se deseja, ou seja, um plano de trabalho antes de começar o desenvolvimento, tornando-o incremental. As sugestões e necessidades do cliente são informadas gradativamente e de acordo com suas necessidades, possibilitando que sempre haja liberação de versões do sistema, sendo tratadas as alterações no momento mais oportuno.

Apesar de abordar de forma mais simplificada, a Metodologia Ágil requer muita disciplina e organização das equipes envolvidas, para se obter as vantagens esperadas com a sua utilização.

A intenção deste novo método não é deixar de lado os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, mas

pretende-se colocá-los em segundo plano, priorizando as iterações e pessoas, com o acompanhamento do *stakeholder* das mudanças que venha acontecer no processo de desenvolvimento do software (Cockburn; Highsmith, 2001).

Nesse contexto, são apresentadas as metodologias ágeis mais conhecidas e utilizadas no mercado atualmente: Scrum e Extreme Programming.

3 Scrum

Scrum não é um processo ou estratégia para conceber produtos, mas um framework que possui muitos processos ou técnicas que podem ser utilizados para torna tudo transparente e visível, possibilitando aos participantes acompanharem o que está sendo realizado ao longo do projeto, permitindo as tomadas de decisões pela equipe durante o desenvolvimento com o objetivo de alcançar melhores resultados ao longo do tempo, para Pressman (2011, pág 95):

> O Scrum é um método de desenvolvimento ágil de software concebido por *Jeff Sutherland e* sua equipe no início dos anos 1990. Os princípios do Scrum são consistentes com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades estruturais: requisitos, análises, projeto, evolução e entrega. Em cada atividade metodológica, ocorrem tarefas a realizar dentro de um padrão de processo chamado *sprint.* O Scrum enfatiza o uso de um conjunto de padrões de processo de software que provaram ser eficazes para projetos com prazos de entrega apertados, requisitos mutáveis e críticos de negócio. Cada um desses padrões define um conjunto de ações, são elas:
> Registro pendente de trabalho (Backlog) – uma lista com prioridades dos requisitos que fornecem valor comercial ao cliente. O gerente avalia o registro e atualiza as prioridades;
> Urgências sprints – consistem em unidades de trabalho solicitadas para atingir um requisito estabelecido no Backlog, de curta duração;
> Reuniões Scrum – são reuniões realizadas diariamente e de forma rápida. Elas ajudam a revelar problemas o mais cedo possível, além disso promove uma estrutura de equipe auto organizada;
> Demos – entrega do incremento de software ao cliente para que a funcionalidade possa ser testada.

Esta metodologia usa essa estrutura de forma iterativa e incremental que funciona da seguinte maneira: no começo de cada iteração (ciclos que podem ser de 15 a 30 dias), a equipe pesquisa qual a tarefa a ser realizada, selecionando o item que concluem que poderá ser um incremento que agregará valor ao produto. A seguir, a equipe trabalha com foco nas funcionalidades preparadas para esta iteração e quando encerrado o ciclo será demonstrado esse incremento aos usuários finais possibilitando a avaliação em ambientes reais e solicitar modificações, se necessárias.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

Segundo Schwaber e Sutherland (2017), o Scrum se apoia em três pilares que controlam o processo de implantação, são eles:

- Transparência: as informações importantes do processo devem ficar sempre a mostra aos responsáveis para que possam analisar os resultados. Estas informações devem ser delineadas por uma padronização comum, para que haja facilidade dos *stakeholders* entenderem e que o restante da equipe de desenvolvimento tenha o mesmo entendimento do que está sendo realizado;
- Inspeção: os elementos do Scrum precisam ser frequentemente verificados na tentativa de encontrar algum tipo de variação. A periodicidade deve ser determinada de maneira que não prejudique a continuação dos trabalhos em curso;
- Adaptação: Logo após a inspeção, se por ventura for identificada que uma característica do processo saiu do curso normal de modo que se nada for feito tornará o resultado final inadmissível, o procedimento de adaptação entre ação para a correção e diminuição das falhas, deixando o produto de acordo com o que foi pensado.

Para Schwaber e Sutherland (2017), os componentes primordiais da estrutura do Scrum são: time Scrum (*Srum Team*), eventos, artefatos e regras.

3.1 Time scrum (*Scrum Team*)

O time Scrum é composto pelo *Product Owner*, o time de desenvolvimento e o *Scrum Master*. Estes times são auto-organizáveis e multifuncionais (SCHWABER; SUTHERLAND, 2017).

A caraterística de serem multifuncionais está liga ao fato da equipe possuir as habilidades para terminar os trabalhos sem criar dependência de outras pessoas que não estão ligadas à equipe. Os times Scrum realizam as entregas do produto de forma iterativa e incremental, e garantem, com isso, com que uma versão do produto esteja sempre funcionando perfeitamente e à disposição do cliente.

As partes de cada membro do time estão descritos nas próximas s:

3.1.1 *Product owner*

O *product owner* é a pessoa que representa os interesses dos usuários e clientes no projeto. É responsável por definir os itens de requisito do projeto numa lista

que se chama Backlog, além do andamento das tarefas ali elencadas. Está sob sua responsabilidade a performance o time de desenvolvimento e de potencializar o valor do produto. Na execução de suas funções, ele deve ordenar os itens do backlog do produto com a finalidade de atingir as metas e também o que foi proposto; definir de maneira minuciosa e de fácil compreensão itens do backlog e ter a certeza que o time de desenvolvimento compreenda tudo; ter certeza que o backlog seja visível, coerente e exato para todos; mudar os requisitos e prioridades, aceitar e rejeitar o resultado de cada *Sprint*, além de demonstrar qual a próxima tarefa do Scrum.

3.1.2 Time de desenvolvimento (*Scrum Team*)

O time de desenvolvimento no Scrum é basicamente compreendido por um grupo de 6 a 10 membros capazes de realizar as atividades e apresentar uma versão do produto quando chegar ao fim cada intervalo estipulado para o desenvolvimento (*Sprint*). O time inteiro trabalha junto, não há na equipe escalonamento de funções baseada em papéis tradicionais, como por exemplo, gerente de projetos, e estão incumbidos de terminar o relatado no backlog a cada iteração.

Como completa Martins (2007), a metodologia Scrum é ideal para projetos dinâmicos e suscetíveis a mudanças de requisitos, sejam eles requisitos novos ou apenas modificados. Entretanto, para aplicação deste método, faz-se necessário entender antes os seus papéis, responsabilidades, conceitos e técnicas das fases de seu ciclo.

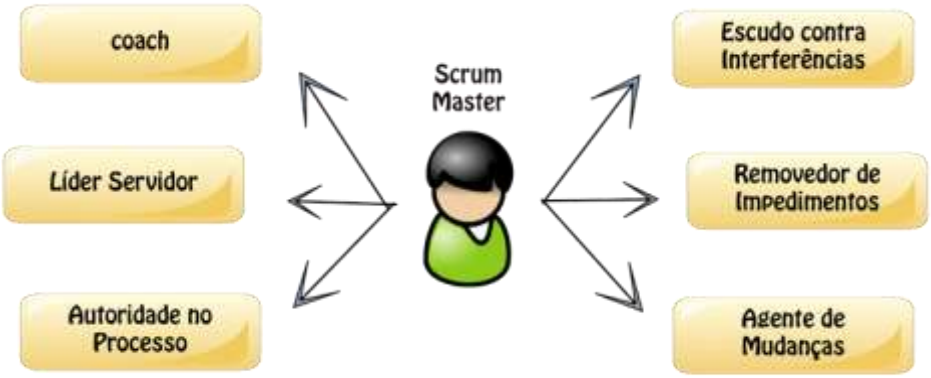
3.1.3 *Scrum Master*

O Scrum Master é encarregado de ter a certeza que o Scrum foi bem assimilado e aplicado pelo time, de modo a garantir que todos estejam engajados ao que foi exposto, isto é, as regras, as maneiras e formas do Scrum. Esta pessoa é um servo-líder para o time Scrum (Schwaber; Sutherland, 2017).

Contribui com o *product owner*, ajudando-o a encontrar técnicas para o gerenciamento do backlog do produto, comunicar claramente a visão, objetivos e itens do backlog do produto ao time de desenvolvimento, compreender e praticar agilidade e facilitar a ocorrência dos eventos Scrum (SCHWABER; SUTHERLAND, 2017).

Para a equipe de desenvolvimento, o Scrum Master é o seu treinador, verificando se o time está em consonância com os preceitos do Scrum em relação ao autogerenciamento e interdisciplinaridade. Remove percalços e procura descomplicar as tarefas do Scrum para o bom andamento dos trabalhos, além de contribuir na elaboração de produtos de alto valor (SCHWABER; SUTHERLAND, 2017).

Figura 1 - Principais responsabilidades do Scrum Master



Fonte: http://www.mindmaster.com.br/scrum-master/.

A Figura 1 mostra as principais responsabilidades do *Scrum Master*, ficando visível a importância de seu papel de liderança e gerenciamento para o sucesso do projeto.

3.2 Eventos Scrum

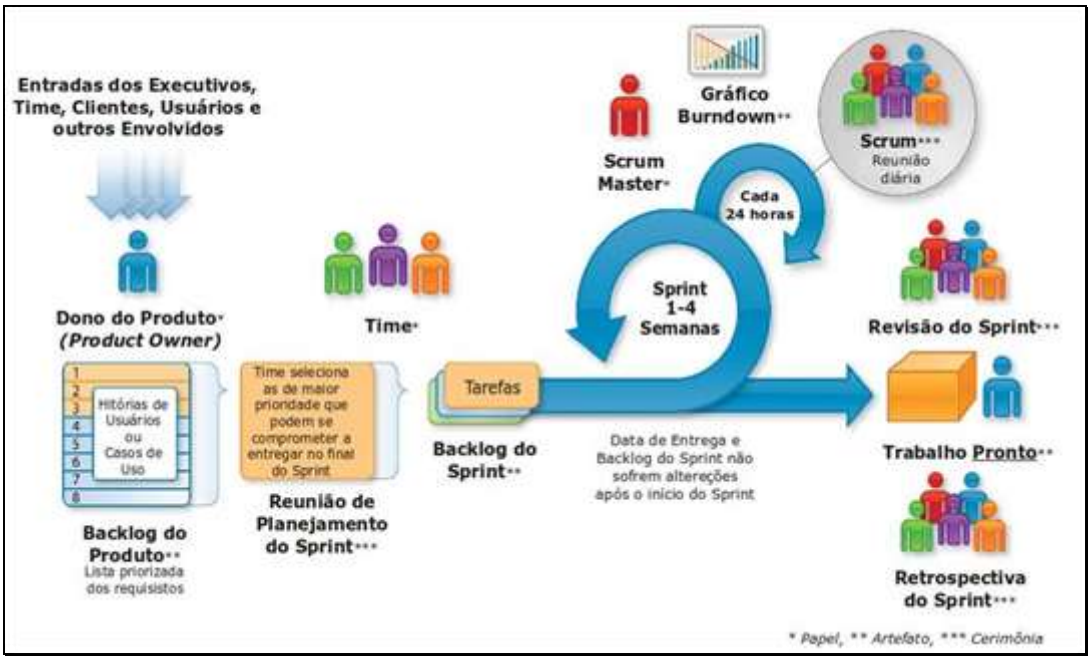
Os eventos são prescritos no Scrum com o objetivo de criar uma rotina e minimizar a necessidade de reuniões não definidas pelo framework. Os eventos Scrum são registrados com data e hora, e para não comprometer o andamento do projeto deve ter uma duração máxima. Cada evento é uma oportunidade de inspecionar e adaptar alguma coisa (SCHWABER; SUTHERLAND, 2017).

3.2.1 *Sprint*

Para Schwaber e Sutherland (2017) é o *sprint* o coração do Scrum (Figura 2), que tem duração média de um mês, que ao seu término é gerada uma versão

potencialmente utilizável do produto para entregar ao cliente. Um novo *sprint* é iniciado imediatamente após o término do anterior.

Figura 2 – Desenvolvimento do Scrum, de forma simplificada.



Fonte: Softhouse (2016).

Em relação ao tempo do *sprint,* Schwaber e Sutherland (2017) sugere que ele tem que ter a duração estável, isto é, se em um *sprint* dura uma semana, a próxima não pode durar três semanas, por exemplo. Isto porque fica difícil organizar uma sequência correta do trabalho e estabelecer o que precisa ser cumprido em um determinado período de tempo (Schwaber; Sutherland, 2017).

O tempo de duração de cada *sprint* deve ser limitado a quatro semanas, porque se ele for muito logo a definição do que está sendo feito pode se alterar, o risco aumentar e caso o tempo seja muito longo a definição do que será construído pode mudar, o risco pode crescer e a complexidade aumentar. O *sprint* poderá ser encerrado, somente pelo *product owner*, antes do prazo do seu termino (Schwaber; Sutherland, 2017).

Durante sua execução nada poderá ser alterado até que se atinja seu objetivo, a qualidade não poderá ser reduzida e o escopo poderá ser esclarecido ou renegociado entre o *product owner* e o time de desenvolvimento, a fim de transmitir mais conhecimento (SCHWABER; SUTHERLAND, 2017).

Para Schwaber e Sutherland (2017), a composição dos *Sprints* é feita por uma reunião de planejamento (*Sprint Planning Meeting*), reuniões diárias (*Daily Sprint*), o

trabalho de desenvolvimento, uma revisão do *Sprint* (*Sprint Review*) e a retrospectiva do *Sprint* (*Sprint Retrospective*).

3.2.1.1 Reunião de planejamento do *Sprint* (*Sprint Planning Meeting*)

A reunião de planejamento é o acontecimento mais importante do Scrum. A finalidade desta reunião é fornecer para a equipe a informação suficiente para que ocorram os trabalhos das semanas vindouras. Por isso é importante que toda a equipe participe da reunião, inclusive o *product owner*. A reunião começa resumindo o objetivo do *Sprint*, e apresentando as funcionalidades que julgar mais importantes à equipe (SCHWABER; SUTHERLAND, 2017).

Assim, em conjunto começam a determinar a estimativa de prazo de cada funcionalidade, sempre iniciando pela mais importante, construindo ao final o *Sprint Backlog*. Se por ventura o tempo estimado não for o que o *product owner* previu, ele poderá alterar a importância da estória no *backlog*. A atuação do Scrum Master nessa reunião é de facilitador (SCHWABER; SUTHERLAND, 2017).

Considerando um *sprint* com duração de um mês, o tempo máximo que uma reunião deve ter é de oito horas. Caso o *sprint* tenha duração menor, este tempo para a reunião deve ser também menor.

3.2.1.2 Reunião Diária (*Daily Scrum*)

A Reunião Diária serve para que o time de desenvolvimento possa inspecionar o trabalho desde a última reunião, organizar as atividades e estabelecer um esboço para o próximo dia, ou até a próxima reunião (SCHWABER; SUTHERLAND, 2017).

Conforme Schwaber e Sutherland (2017), o *product owner* deve garantir que a reunião aconteça e que tão somente quem faz parte do time deve participar. Deve ter a duração de quinze minutos apenas, e deve ser feita sempre no mesmo horário e mesmo local para diminuir a complexidade. Nesta reunião, são realizadas três perguntas para a equipe de desenvolvimento que devem responder, são elas:

- O que fiz ontem que ajudou a atender a meta do *sprint*?
- O que farei hoje para ajudar a atender a meta do *sprint*?
- Eu vejo algum obstáculo que impeça o atendimento da meta do *sprint*?

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

O objetivo da reunião diária é tornar de conhecimento da equipe os acontecimentos do dia anterior, identificar impedimentos e priorizar o trabalho que será realizado durante o dia. Todo o problema apresentado durante essa reunião será tratado em uma outra reunião, com um grupo reduzido de membros.

Nesta reunião, o Scrum Master tem o papel de controlar como as tarefas devem ser executadas e garantir que não haja interferência externa. Assim, ele blindará o time de possíveis desvios do objetivo. São admitidas participação de membros externos somente como ouvintes e não podem opinar (Schwaber; Sutherland, 2017).

3.2.1.3 Trabalho de desenvolvimento

Trata-se do desenvolvimento da funcionalidade selecionada na reunião de planejamento para ser entregue pela equipe de desenvolvimento ao final do sprint.

3.2.1.4 Revisão do *Sprint* (*Sprint Review*)

Acontece sempre no fim do *sprint,* sendo programada para que não ultrapasse quatro horas. Nesta fase, o time de desenvolvimento somente poderá apresentar aquilo que foi totalmente concluído, explanando o que foi feito na duração do sprint. O produto gerado é inspecionado tendo como parâmetro os objetivos que foram definidos para aquele *Sprint*, conta com a presença do *product owner* (Schwaber; Sutherland, 2017).

3.2.1.5 Retrospectiva do *Sprint* (*Sprint Retrospective*)

Logo após a equipe demonstrar o que foi feito no *sprint* anterior, este momento serve para que eles reflitam sobre o poderia melhorar e o que pode ser feito de melhor no próximo *sprint*. Tão somente serve para analisar o processo e não encontrar culpados. Por esse motivo a equipe deve demonstrar a sua maturidade emocional (Schwaber; Sutherland, 2017).

Para isso é necessário que cada membro da equipe assuma a responsabilidade do que fez. Além disso devem apontar os problemas que atormentam eles. Por outro lado, os outros membros da equipe devem ouvir e encontrar uma solução para a situação apresentada (Schwaber; Sutherland, 2017).

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

3.3 Artefatos

Para Schwaber e Sutherland (2017) os artefatos do Scrum representam trabalho ou valor para fornecer transparência e oportunidades para inspeção e adaptação. Artefatos definidos pelo Scrum são projetados especificamente para maximizar transparência das principais informações para que todos tenham o mesmo entendimento.

3.3.1 Backlog do Produto (*Product Backlog*)

Para Schwaber e Sutherland (2017), o backlog é definida como uma lista que descreve tudo aquilo que deve estar contido no produto. Ela é a fonte dos requisitos para qualquer alteração que possa ser pleiteada. O *product owner* lista todas as características, funções, requisitos, melhorias e correções. A sequência dos itens no *backlog* obedece a uma ordem de prioridades. As funcionalidades mais importantes devem ser colocadas no topo da lista. Para mensurar a prioridade são analisados os fatores como valor do produto, custo, risco e o conhecimento.

O *product owner* é a pessoa responsável por ordenar e gerenciar a sequência do backlog do produto e demonstrá-lo através de uma lista de prioridades. Para a composição dele pode haver a participação de todas as partes envolvidas no projeto, principalmente a equipe de desenvolvimento (SCHWABER; SUTHERLAND, 2017).

Figura 3 – Backlog do Produto

Fonte: http://www.mindmaster.com.br/scrum/.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

A Figura 3 exemplifica a estrutura *product backlog*, é um registro que está em constante desenvolvimento, podendo ter algo adicionado, corrigido e retirado, dependendo da análise da equipe. No topo são agrupadas as funcionalidades com maior prioridade para direcionamento do trabalho.

3.4 Regras

O framework Scrum consiste nos times Scrum e suas funções associadas aos eventos, artefatos e regras. Cada componente dentro da estrutura serve a um propósito específico e é essencial para sucesso e uso do Scrum. As regras do Scrum unem as funções, eventos e artefatos, governando os relacionamentos e interação entre eles (SCHWABER; SUTHERLAND, 2017).

As regras do Scrum orientam para seguir o papel de cada item descrito no documento elaborado por Schwaber e Sutherland (2017).

4 Programação Extrema (XP)

Segundo Beck (2008), a *Extreme Programming* (Programação Extrema) é considerada uma metodologia leve, eficiente e de baixo risco, com forma flexível, previsível, científica. Ela é indicada para times de desenvolvimento de pequeno e médio tamanhos, nos quais os requisitos são incertos ou se alteram com frequência. Provavelmente a XP é a metodologia ágil que mais se conhece e se utiliza. A grande quantidade de adeptos demonstra a sua aceitação. Isto pois oferece maneiras para que os desenvolvedores atuem mais rápidos com as eventuais mudanças do projeto, ainda que este esteja na sua fase de finalização.

Beck (2001) define o XP como uma disciplina de desenvolvimento de software, e para que sua aplicação seja implementada se faz necessário seguir todas regras que o XP recomenda. É inconcebível escolher ou não as regras. Porque se elas não estiverem sendo seguidas não há o que se dizer em metodologia XP. Ele foi criado para atuar em projetos desenvolvidos por equipes de dois a dez programadores, e onde um trabalho razoável de execução de testes pode ser feito em pelo menos um dia.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

Neste modelo é necessário a presença constante do cliente ou usuário final do produto, pois eles atuam nas tomadas de decisões, esclarecendo dúvidas quanto aos requisitos ou que possam surgir no transcorrer das atividades.

Para Pressman (2011), o XP é um conjunto de cinco valores que estabelecem as bases para todo trabalho realizado como parte: comunicação, simplicidade, feedback (realimentação ou retorno), coragem e respeito. Cada um desses valores é usado como um direcionador das atividades, ações e tarefas específicas do XP.

Abaixo descreve-se estes valores com o intuito de alcançar a eficiência e a competência no decorrer dos trabalhos de desenvolvimento:

- Comunicação: O XP salienta que deve haver estreitamento de relações entre os desenvolvedores e clientes. A comunicação, embora informal, deve fluir de maneira rápida e eficiente e a característica desta metodologia emprega muitas práticas que não podem ser realizadas sem a comunicação. Além do mais, muitos problemas em processos de desenvolvimento acontecem devido à falta de comunicação entre os integrantes da equipe, por desleixo ou esquecimento.
- Simplicidade: O XP restringe a fazer com os desenvolvedores projetam aquilo que é imediato sem preocupações com o que deve ser feito no futuro. O intuito é fazer o mais simples e futuramente, se necessário, melhorar, pois o tempo gasto com tarefas complexas podem nunca serem utilizadas. Simplicidade e comunicação tem uma proximidade muito importante, pois quanto mais comunicação existir, mais compreensão do que necessita ser realizado e mais simples se torna o projeto.
- Feedback: O feedback vem de três formas: do cliente, dos desenvolvedores e do próprio software. Quanto mais rápido ele acontecer, mais rápido a equipe perceberá se os trabalhos estão seguindo para a direção certa, tornando mais fácil a correção de eventuais falhas.
- Coragem: Segundo Pressman (2011), uma palavra melhor poderia ser disciplina. Isto pois há uma pressão para elaboração de projetos pensando em futuros requisitos. A equipe XP ágil deve ter disciplina e coragem para projetar para hoje, reconhecendo que se considerar as necessidades futuras poderá colocar em jogo toda a agilidade do processo.
- Respeito: O respeito aqui considerado está ligado tanto aos membros da equipe, clientes, usuários finais além do software desenvolvido. Através das entregas

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

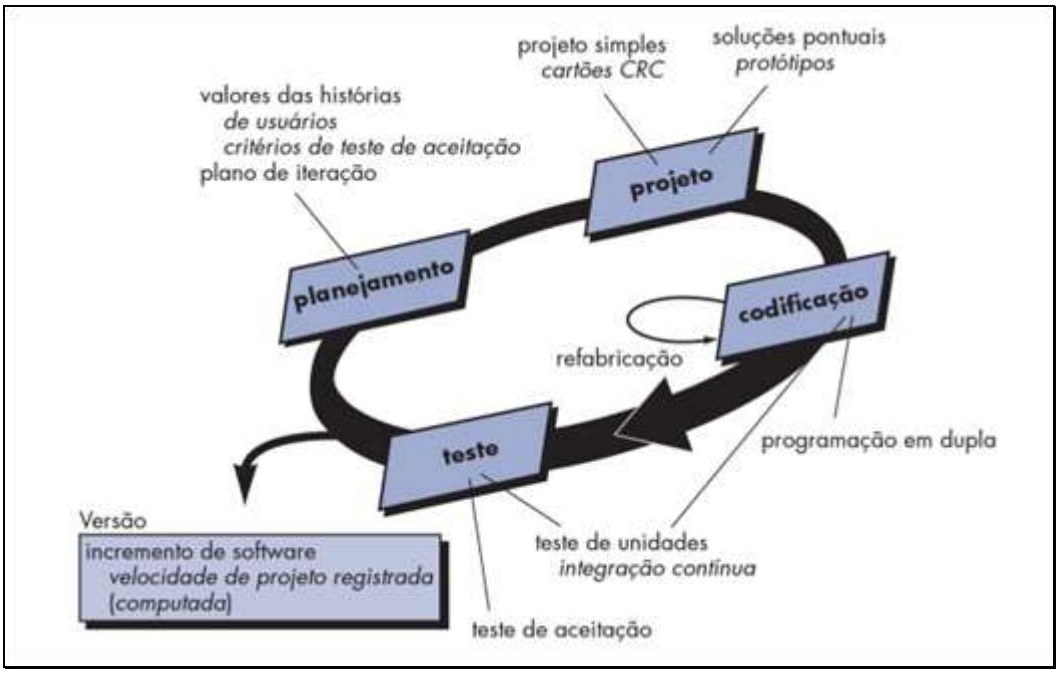
que acontecem paulatinamente vai-se criando cada vez mais respeito pelo método ágil.

Segundo Pressman (2011), a Extreme Programming emprega uma abordagem orientada a objetos como seu paradigma de desenvolvimento preferido e envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas:

- Planejamento
- Projeto
- Codificação
- Teste

Na Figura 4 tem-se uma visão bastante elucidativa do funcionamento dos processos na metodologia XP.

Figura 4 - O processo da Extreme Programming (XP)



Fonte: Pressman(2011).

Verifica-se na Figura 4, que o planejamento começa com elicitação de requisitos que tem característica informal de ouvir o cliente para se que consiga uma percepção ampla do projeto. Nesta fase são escritos os cartões de histórias (user stories) que usualmente usam o padrão de três sentenças: o resultado, a característica e a funcionalidade requisitada.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

Os cartões de histórias são escritos pelos clientes em uma ficha, que atribuem um valor baseado no negócio. Em seguida a equipe avaliam e atribuem um custo, medidos em tempo (dias de programação). Caso a estimativa de tempo ultrapassar três semanas é solicitado ao cliente para dividir o cartão em dois ou mais outros cartões. Há de se considerar aqui que os cartões podem ser escritos a qualquer momento do desenvolvimento do projeto. O tempo adequado é aquele suficiente para transformar a história em código, sem distrações e sabendo exatamente o que irá fazer. Deve-se incluir neste tempo a realização de testes também. A equipe de negócios deve definir o escopo, prioridade, composição das versões e datas de entrega. Isto baseados nas informações que equipe de desenvolvimento lhes fornecerem.

O projeto pode ser avaliado pelo escopo: o quanto deve ser feito; recursos: quantas pessoas estão disponíveis; tempo: quando o produto ou uma versão dele será entregue; qualidade: quão bom será o software e o quanto dele foi testado.

Na etapa do projeto, segue rigorosamente o princípio KIS (*keep it simple*, isto é, preserve a simplicidade), optando sempre por um projeto simples do que uma representação mais complexa. O foco está na simplicidade do código para diminuir o tempo que será gasto no desenvolvimento e eventuais mudanças futuramente. Adotar a prática de que quando for detectado algo complicado deve trocá-lo por um mais simples (PRESSMAN, 2011).

A equipe deve ater-se somente ao que foi definido para o dia, nunca se deve implantar funcionalidades adiantada, porque além de quebrar o planejado, elas podem nem ser utilizadas. O conceito de refatorar pode ser utilizado em todo o momento do desenvolvimento e isto gera economia de tempo e aumento de qualidade. O código deve ser sempre claro e enxuto pois isso deixa ele mais fácil de compreender, alterar e aumentar. Refatorar significa que o “projetar” é feito de forma contínua enquanto o software estiver em elaboração.

Na etapa de codificação, a equipe não passa imediatamente para a codificação do sistema, passa a realizar uma série de testes de unidades que exercitarão cada uma das histórias a ser inclusas na versão corrente.

> Uma vez criado o teste de unidades, o desenvolvedor poderá melhor focar-se no que deve ser implementado para ser aprovado no teste. Nada estranho é adicionado. Estando o código completo, este pode ser testado em unidade imediatamente, e, dessa forma, prover, instantaneamente, feedback para os desenvolvedores. (PRESSMAN, 2011, pág 88.)

Como padrão, a atividade de codificação é realizada em duplas. Isto é, duas pessoas utilizando o mesmo computador, o que gera mais qualidade ao produto. Integrar frequentemente evita conflitos no código e o fracionamento do desenvolvimento, característicos de pessoas que não se comunicam sobre o que estão fazendo. O fato de utilizar um computador por dupla garante estabilidade pois pode ocorrer somente uma liberação de versão. A coletividade neste caso estimula a equipe a contribuir com ideias para o andamento do projeto.

Na última etapa, os testes de unidade criados devem ser implementados usando-se uma metodologia que os capacite a ser automatizados (assim, poderão ser executados fácil e repetidamente). Todo código deve ter sua unidade de teste. Toda vez que novas funcionalidades forem acrescentadas deve começar uma nova bateria de testes, na qual os testes devem abranger os códigos antes de qualquer entrega. Assim tem-se a certeza que o sistema está sempre funcionando. Quando um problema é encontrado faz-se uma rotina de testes para que este problema não volte a acontecer (PRESSMAN, 2011).

Os testes dos clientes, também conhecido com testes de aceitação da XP, são realizados nas reuniões de planejamento da iteração seguinte. Como base para estes testes são utilizados os cartões de histórias. Uma história não está completa se não vencer o teste de aceitação. Neste modelo, os clientes são os responsáveis por verificar a exatidão, verificar se tudo está conforme planejado ou apontar falhas. Os testes de aceitação são obtidos de histórias de usuários implementadas como parte de uma versão do software (PRESSMAN, 2011).

5 Comparação da metodologia tradicional e metodologia ágil

Segundo Cockburn e Highsmith (2001), a maioria das metodologias ágeis não possuem nada que seja novidade em relação as metodologias tradicionais, mas o que as diferenciam são o enfoque e os valores. Enquanto o enfoque das metodologias ágeis está nas pessoas envolvidas e não nos processos e algoritmos como as metodologias tradicionais. Além de que há a preocupação de gastar o menor tempo possível com a documentação do que com a implementação do software.

Com a característica de serem adaptativas as metodologias ágeis têm sido amplamente utilizadas, principalmente com o fato que os requisitos ou funcionalidades dos sistemas sofrem constantes alterações ao longo da sua implementação,

Revista EduFatec: educação, tecnologia e gestão
V.1 N.1 janeiro-junho/2018

demonstrando ser um contraponto em relação as metodologias tradicionais, em que cada etapa deve ser criteriosamente analisada e validada para iniciar o passo seguinte. A análise prévia da metodologia tradicional resulta na elevação dos custos, pode comprometer a qualidade do software e no prazo final de entrega do produto.

Embora as metodologias ágeis estejam sendo aprimoradas e com pouco tempo de utilização, um artigo (CHARETTE, 2001) comparando as metodologias mostrou que os projetos usando os métodos ágeis obtiveram melhores resultados em relação ao cumprimento de prazos, de custos e padrões de qualidade. Neste mesmo estudo mostra que há crescimento do tamanho dos projetos e das equipes que utilizam metodologias ágeis.

Segundo Soares (2004), apesar do crescente interesse pelas metodologias ágeis, ainda faltam comprovações do seu uso em projetos grandes e críticos, e quanto mais organizações usarem essas metodologias, melhores serão os resultados empíricos em termos de vantagens e desvantagens, riscos e procedimentos para sua adoção nas organizações. Mesmo assim, apresentam promissores resultados em relação a qualidade, confiança, datas de entrega e custos.

Considerações finais

Neste trabalho foram abordadas as metodologias ágeis, com a apresentação dos modelos Scrum e XP, que dentre as metodologias existentes e utilizadas atualmente no mercado, foram escolhidas pela sua grande aceitação entre as empresas de desenvolvimento. Além de atingir seu objetivo ao demonstrar que a utilização de metodologias ágeis tem sido mais vantajosa para o gerenciamento e a execução dos projetos de desenvolvimento de software, observando que a escolha do método deverá levar em conta as especificidades do projeto e como ocorrerá relacionamento com o cliente.

Contribui ainda ao esclarecer que na contratação para desenvolver um projeto de software, o cliente poderá optar por ter uma documentação detalhada do produto obtida com a metodologia tradicional, ou um sistema desenvolvido com agilidade com entregas por partes obtido com a utilização das metodologias ágeis.

Em relação os métodos Scrum e XP, pode-se pensar em uma metodologia ágil híbrida, na qual utiliza-se o que cada um tem de melhor, isto é, pode-se seguir toda a

recomendação do Scrum e realizar práticas como programação em pares, desenvolvimento guiado por testes, entre outras do XP.

Referências

AGILE ALLIANCE, **Agile aliance**: The Agile Manifesto, 2013. Disponível em: <https://www.agilealliance.org/agile101/the-agile-manifesto> Acessado em 28 jun. 2018.

BECK, Kent et al. **Manifesto for agile software development**. 2001.

BECK, K. (2008) **Programação extrema (XP) Explicada**. Porto Alegre: Bookman.

COCKBURN, A. e HIGHSMITH, J. "**Agile Software Development: The Business of Innovation**", IEEE Computer, pp. 120-127, Set. 2001. Disponível em: <https://ieeexplore.ieee.org/abstract/document/947100/> Acessado em 28 jun. 2018.

COHN, Mike. **Desenvolvimento de Software com Scrum**: Aplicando métodos ágeis com sucesso, Bookman, Porto Alegre, 2011.

MARTINS, José Carlos Cordeiro. **Técnicas Para Gerenciamento de Projetos de Software**. Rio de Janeiro. Brasport, 2007.

PRESSMAN, Roger S. **Engenharia de Software**: Uma abordagem profissional, Bookman, Porto Alegre, 2011;

SCHWABER, Ken; SUTHERLAND, Jeff. ***The Scrum Guide****: The Definitive Guide to Scrum: The Rules of the Game*. 2013. Disponível em: <http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100>. Acesso em: 28 jun. 2018.

SOARES, Michel dos Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**. INFOCOMP, [S.l.], v. 3, n. 2, p. 8-13, nov. 2004. Disponível em: <http://www.dcc.ufla.br/infocomp/index.php/INFOCOMP/article/view/68>. Acesso em: 09 jul. 2018.

SOMMERVILLE, Ian. **Engenharia de Software**, Person, São Paulo, 2010.