METODOLOGIAS DE DESENVOLVIMENTO ÁGEIS DE SOFTWARE SCRUM E EXTREME PROGRAMING

AGILE SOFTWARE DEVELOPMENT METHODOLOGIES SCRUM AND EXTREME PROGRAMING

Marcio Francisco Nogueira[*]
Mauro Zamaro[**]

RESUMO

Este trabalho consiste em fazer um comparativo entre as metodologias ágeis de desenvolvimento, sendo estas Scrum, Programação extrema (Extreme Programming-XP), o trabalho trará consigo uma descrição dos métodos de desenvolvimento assim como um comparativo dentre eles citando suas principais vantagens e desvantagens. Hoje em dia, é comum vermos vários projetos serem fracassados ou porque foram cancelados, atingiram um orçamento inviável ou que foram entregues fora dos prazos estipulados, os métodos de desenvolvimento ágil visam uma maior dinâmica e eficiência na realização das tarefas com prazos e objetivos pré-estipulados.

Palavras-chave: Agile Alliance. Scrum. Metodologia Ágil. Extreme Programming. Manifesto Ágil.

ABSTRACT

This work intends to make a set of comparisons among agile development methodologies as SCRUM and Extreme Programming bringing its descriptions and major of its involved concepts. The focus of this work is the comparision of its advantages and disadvantages and try to find out the common behaviors on failed projects - late projects, over-budgeted projects or cancelled projects - based on the project management methodologies concepts that tries to bring more efficiency and dynamism to development process and avoid its failures.

Keywords: Agile Alliance. Scrum. Agile Methodologies. Extreme Programming. Agile manifest.

Introdução

O desenvolvimento ágil de software é uma nova maneira de desenvolver softwares a qual se diferencia das metodologias tradicionais, tendo esta como sua maior prioridade atender o cliente, dentro dos prazos estipulados. Ele tem seu foco nas

[*] Graduado em Ciência da Computação pela Faculdade de Tecnologia, Ciências e Educação (FATECE). marciofnogueira@hotmail.com
[**] Acadêmica da Força Aérea (AFA/Pirassununga-SP).

pessoas, em um ambiente aonde os requisitos surgem e mudam rapidamente, pode-se dizer que é uma metodologia que tem por objetivo gerenciar o desenvolvimento de um software.

À medida que as organizações tornam-se cada vez mais dependentes da indústria do *software*, ficam explícitos os problemas relacionados ao processo de desenvolvimento de softwares: alto custo, alta complexidade, dificuldade de manutenção. No cenário atual, os softwares são cada vez mais complexos, contendo várias funcionalidades, o mercado está cada vez mais exigente quanto à qualidade do sistema que estão adquirindo, assim como os prazos de entrega do produto.

A indústria de software e seus desenvolvedores trabalhavam seguindo metodologias de desenvolvimento hoje tidas como muito retroativas por terem seu maior enfoque na parte de documentação de todo o projeto, sendo elas respectivamente o modelo waterfall (cascata) e o modelo espiral.

Em virtude destas dificuldades, em 2001 reuniram-se uma equipe de renomados desenvolvedores tendo nomes como Kent Beck e Ken Schwaber, e em conjunto criaram o manifesto ágil, com princípios focados em resultados rápidos e entregas constantes, diversas metodologias se consolidaram, como Crystal, Extreme Programming (XP), Scrum, dentre outras de menor expressão, as quais não abordaremos neste trabalho.

Neste trabalho estaremos abordando as duas metodologias ágeis de maior expressão na atualidade, sendo a primeira delas o Scrum que é voltado ao gerenciamento de todo o projeto e desenvolvimento de software, e também o Extreme Programming (XP), que é voltado especificamente às práticas de programação.

1 O Manifesto Ágil de Desenvolvimento de Software

O movimento ágil tem como seu marco inicial o Manifesto Ágil, o qual foi proposto em meados de 2001 nos Estados Unidos, por um grupo de especialistas em desenvolvimento de software. Eles uniram com intuito de buscar novas formas para melhorar a velocidade de desenvolvimento, evitando processos demorados e burocráticos dos métodos tradicionais, definiram valores e princípios que permitissem a equipe de desenvolvimento produzir e responder de forma rápida as alterações.

Tendo como seus principais valores:

> Indivíduos e interação entre eles mais que processos e ferramentas;

> Software em funcionamento mais que documentação abrangente;
> Colaboração com o cliente mais que negociação de contratos;
> Responder a mudanças mais que seguir um plano.
> (MANIFESTOAGIL)

1.1 O que são metodologias ágeis

A metodologia Ágil é uma abordagem da gestão de projetos utilizados no desenvolvimento de softwares, utilizando uma abordagem de planejamento e execução interativa e incremental, voltados a processos empíricos (imprevisíveis, complexos, com várias incertezas, repetitivos e com muitas alterações ao longo do projeto).

Seguindo os princípios das metodologias ágeis[1], elas trabalham para o aumento da produtividade no desenvolvimento de software através de atividades pertinentes com seus valores. O objetivo é ter um software funcionando o mais rápido possível, estando este o mais completo possível em quaisquer etapas do projeto.

A metodologia ágil engloba mudanças em quaisquer fases do projeto, pois ela subentende que o software nunca está completo, e o cliente sempre sugere novas alterações.

2 Apresentação das metodologias a serem analisadas

O enfoque deste artigo é explanar as duas metodologias com maior enfoque no mercado, inicialmente será mostrado os princípios básicos das metodologias Scrum e XP, respectivamente, com intuito de buscar um melhor entendimento de ambas.

2.1 O que é Scrum

O Scrum desponta-se como uma metodologia extremamente flexível, tendo seus princípios nas boas práticas de gestão, ele se aplica tanto em projetos grandes quanto a pequenos.

Um dos principais objetivos do Scrum é conseguir uma avaliação correta do ambiente em sua evolução, adaptando-se a quaisquer modificações, este método apenas

[1] Com o intuito de auxiliar as pessoas a compreenderem melhor o desenvolvimento de software ágil, em 2001 os membros da Agile Alliance apuraram o enunciado do Manifesto Ágil, criando assim os doze princípios que as metodologias ágeis devem seguir.

estabelece um conjunto de regras para a gestão que devem ser adaptadas para o êxito do projeto.

O Scrum foi desenvolvido por Ken Schwaber e Jeff Sutherland e vem sendo utilizado para gerenciar o desenvolvimento de produtos complexos desde 1990, tal método nasceu da necessidade de encontrar uma metodologia que abordasse os problemas do desenvolvimento de software de uma forma que não fosse a tradicional. Seus criadores fizeram sua primeira co-apresentação na conferência de OOSPSLA, em 1995.

O Scrum é um processo iterativo e incremental de desenvolvimento ágil de software, ou seja, é um método de gerenciamento de projetos de softwares e de produtos ou desenvolvimento de aplicações. O Scrum não só reforçou o interesse em gerenciamento de projetos, mas também desafiou as ideias convencionais sobre essa gestão.

Ele caracteriza-se por ser um processo ágil de desenvolvimento que permite manter o foco na entrega do maior valor de negócio com o menor tempo possível, desta forma ele permite de forma rápida e continua uma inspeção do produto em desenvolvimento.

A figura abaixo demonstra o ciclo de vida do Scrum:

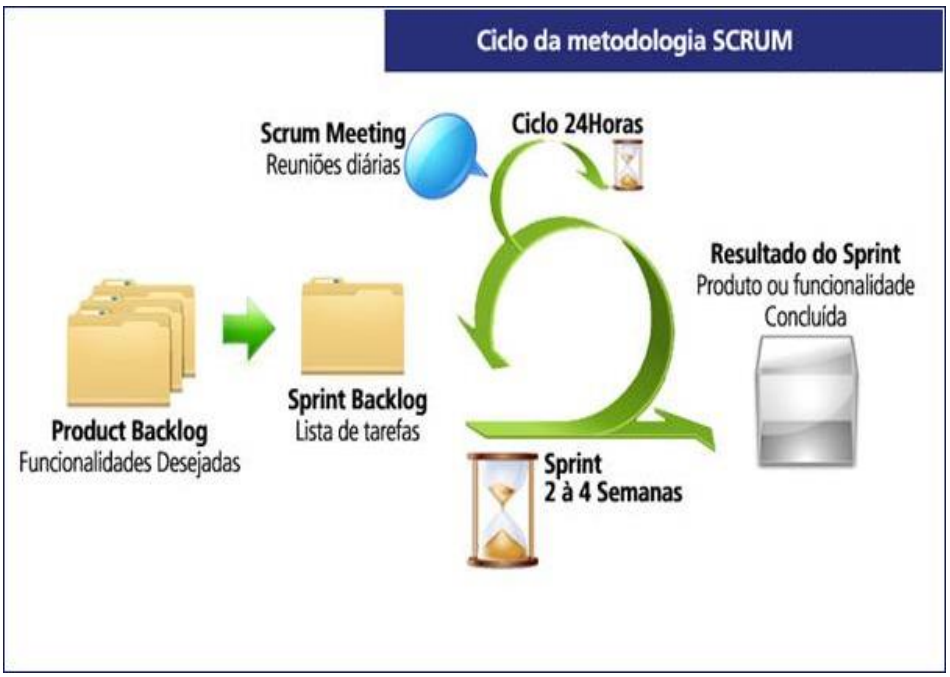


Figura 1 - Ciclo Metodologia Scrum
Fonte: http://www.dadosweb.com.br/wp-content/uploads/scrum.jpg

A Sprint é a unidade básica de desenvolvimento em Scrum, ela tem um período de duração de quatro semanas, ou seja, um time Box, durante este período o projeto ou parte dele é desenvolvida, com o término de uma Sprint sempre se inicia uma nova.

Cada Sprint é precedida por uma reunião de planejamento, onde as tarefas para o Sprint são identificadas e um compromisso estimado e um objetivo para a Sprint é feita.

2.2 Principais Papéis no Scrum

A organização dos recursos em um time Scrum é dividida em três papéis e responsabilidades, sendo estas o Product Owner, o Scrum Master e a Equipe de desenvolvimento.

2.2.1 Product Owner

O Product Owner ou dono do produto é a pessoa responsável por gerenciar o Product Backlog, ele também é incumbido de maximizar o valor do produto e do trabalho realizado pela equipe em vista que tem uma visão do produto em vários níveis, ele pode ser o próprio cliente ou uma pessoa que represente a função deste. Dentro do Scrum o Product Owner tem sua importância tão grande quando o Scrum Master ou a própria equipe de desenvolvimento.

2.2.2 Scrum Master

O Scrum Master é a pessoa responsável por garantir o entendimento e aplicação do Scrum, o mesmo faz garantir que o time de desenvolvedores aderiram à teoria, regras e práticas do Scrum, de forma que o time venha a tornar-se funcional e produtivo e possa acompanhar o que vem sendo realizado. Ele trabalha próximo ao Product Owner e auxilia a equipe removendo todos e quaisquer tipos de impedimento que possa a vir ocorrer nas Sprints.

2.2.3 Equipe (Team)

É o conjunto de pessoas que fica responsável pelo desenvolvimento das Sprints, os times de Scrum entregam os produtos de forma iterativa e incremental, entregas realizadas de forma incremental garantem que uma versão funcional do produto esteja sempre disponível.

O modelo de equipe no Scrum é projetado de forma que venha aperfeiçoar a produtividade, criatividade e flexibilidade.

2.3 Artefatos

Na metodologia Scrum temos quatro ferramentas básicas para trabalhar, são elas o Product backlog, Sprint Backlog, User Stories e Gráfico de Burndown.

2.3.1 Product backlog

O Product Backlog é uma lista de tudo que será necessário ao produto, ele é dinâmico, isto é, muda constantemente para identificar o que o produto necessita para tornar-se mais apropriado, competitivo e útil, o Product Owner fica responsável pelo Product Backlog. É importante ressaltar que o Product Backlog nunca está completo, o mesmo evolui da mesma forma que o produto e o ambiente onde será utilizado, ele existirá enquanto o produto existir.

2.3.2 Sprint backlog

O Sprint Backlog é um conjunto de tarefas que a equipe de desenvolvimento se compromete a fazer em um Sprint, tais itens são retirados do Product Backlog, tendo as suas prioridades definidas pelo Product Owner. Através do Sprint Backlog é definido qual trabalho será realizado para converter o Product Backlog em um incremento.

O Sprint Backlog vai surgindo durante a própria Sprint, podendo ser modificado durante toda a Sprint, sempre que é necessário um novo trabalho a equipe de desenvolvimento adiciona este ao Sprint Backlog.

Sempre que um trabalho é concluído as estimativas deste são atualizadas, e quando os elementos são considerados como desnecessários estes são removidos. Somente a equipe de desenvolvimento pode modificar o Sprint Backlog.

2.3.3 User Stories

As User Stories são uma simples discrição de uma característica contada a partir da perspectiva da pessoa que deseja um novo recurso, normalmente um usuário ou cliente do sistema. Normalmente este requisito é capturado em um parágrafo onde descreve a necessidade do usuário de forma sucinta, utilizando uma linguagem comum ao negócio.

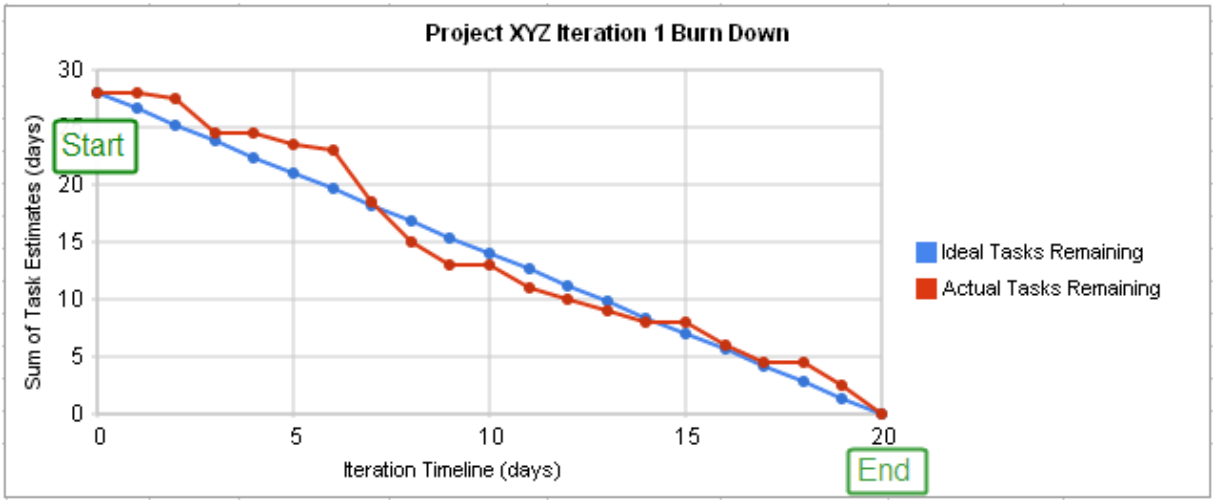
Muitas equipes de desenvolvimento Scrum adotaram o modelo de User Stories proposto por Mike Cohn, onde se identifica quem é o usuário final, o que o mesmo deseja, pondo isso em uma única etapa, este modelo de User Stories é frequentemente descrito pelo autor desta forma: "Como [o papel do usuário final], eu quero [objetivo] para que [a razão]".

As User Stories são escritas em fichas ou notas e armazenadas em uma caixa ou em quadros fixos nas paredes, visando facilitar o planejamento e discussão.

2.3.4 Gráficos de Burndown

O gráfico de Burndown é um dos elementos mais importantes para o monitoramento do progresso da equipe de desenvolvimento da equipe ágil. O eixo vertical (y) representa a quantidade de trabalho a ser realizado ainda já o eixo horizontal (x) representa o tempo.

Gráfico 1 – Gráfico de Burndown



Fonte: http://en.wikipedia.org/wiki/Burn_down_chart

A unidade de tempo dada pelo eixo y pode ser em horas, dias, semanas ou em sprints, já a unidade no eixo x pode ser em horas trabalhadas ou pontos, é importante ressaltar que a unidade escolhida seja a mais adequada a cada projeto ou empresa, sendo que cada unidade foca um objetivo diferente. Através do gráfico de Burndown é possível notarmos o andamento do projeto e o que pode ser melhorado.

2.4 Extreme Programming (XP)

Extreme Programming é uma metodologia de desenvolvimento ágil de software que nasceu nos Estados Unidos em meados da década de 90, e tem ganhado muitos adeptos ao redor do mundo por ajudar a criar e desenvolver sistemas de melhor qualidade com um tempo relativamente menor em relação às metodologias convencionais.

O XP é organizado em ciclos curtos de feedback a fim de que os usuários possam solicitar novas funcionalidades e aprenderem sobre as já concluídas através do software já implementado, a equipe só segue adiante caso o resultado esteja correto, caso ocorra falhas a equipe corrige-as antes de prosseguir para a próxima funcionalidade. Este processo engloba a priorização de poucas funcionalidades a serem implementadas e a simplificação das mesmas.

Um dos principais objetivos é a apresentação das funcionalidades para o usuário a fim que se possam identificar possíveis falhas de forma mais precoce e efetuar as correções necessárias

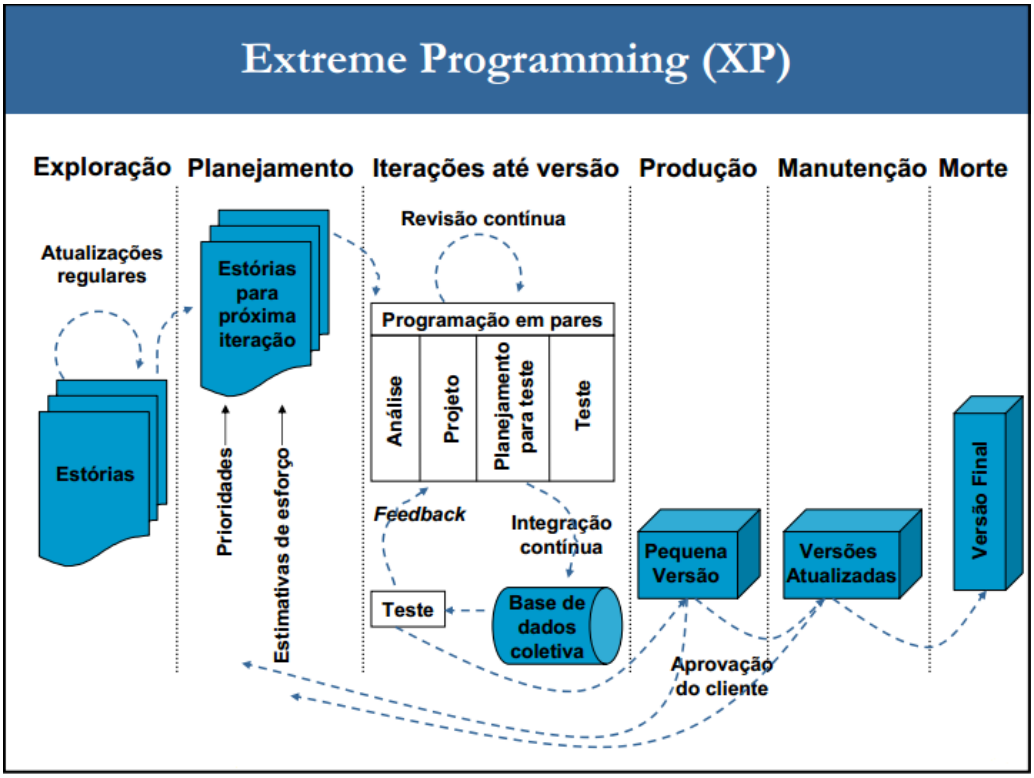


Figura 2 - Ciclo de Vida Extreme Programming
Fonte: ttp://www2.dc.ufscar.br/~junia/MetAgEds.pdf

2.4.1 Valores

O XP adota quatro valores que servem como parâmetro para guiar as equipes envolvidas no processo de desenvolvimento e criação de software, ou seja, eles definirão as atitudes das equipes e suas principais prioridades, sendo eles respectivamente: comunicação, feedback, simplicidade e coragem.

2.4.2 Comunicação

É obrigatória para que não existam brechas em processos e problemas entre a equipe, cliente e fornecedor, ela foca em conseguir um entendimento pessoa-a-pessoa sobre o problema, utilizando o máximo de iteração entre a equipe de desenvolvimento e o mínimo de documentação. Tal prática tende estimular a comunicação entre os gerentes, programadores e clientes.

2.4.3 Feedback

O feedback é muito importante para que as pessoas envolvidas aprendam cada vez mais sobre o sistema corrigindo erros e melhorando-o, também pode-se dizer que é a prática fundamentada de retornar informações entre os membros da equipe e na relação com o cliente. Os programadores obtêm um retorno sobre a lógica dos programas escrevendo casos e executando os testes.

2.4.4 Simplicidade

O XP afirma que se deve fazer um bom trabalho hoje (comunicação, testes, refatoração), para resolver os problemas de hoje, e confiar na sua habilidade de inserir ações mais complexas no futuro caso haja necessidade, ou seja, desenvolvendo coisas simples e funcionais, seguindo este intuito o cliente terá a funcionalidade o mais breve possível e da forma desejada, dando um feedback de forma instantânea. O desenvolvedor irá apenas implementar o necessário para que o cliente tenha sua solicitação atendida.

2.4.5 Coragem

O XP é uma metodologia que se baseia em vários princípios tais como:

- Desenvolver software de forma incremental;
- Manter o sistema simples;
- Permitir que o cliente priorizasse as funcionalidades;
- Fazer que os desenvolvedores trabalhassem em pares;
- Integrar o sistema diversas vezes ao dia.

Por conta disso é dito que a equipe precisa ter coragem para adotar tais ações.

O XP também define um conjunto de valores que devem ser seguidos pela equipe. Tais princípios irão servir para ajudar na escolha de alternativas para soluções dos problemas durante o andamento do projeto. Tendo como principais princípios: feedback rápido, assumir simplicidade, mudança incremental, abraçando mudanças e trabalho de qualidade.

2.5 Práticas

Mesmo sendo uma metodologia dinâmica e flexível o XP exige muita disciplina para utilizá-la, nesse capítulo iremos abordar algumas práticas essenciais para o bom andamento de um projeto.

2.5.1 Planejamento

No início de cada semana é realizada uma reunião de planejamento onde o cliente e o desenvolvedor participam, nesta reunião são definidas as prioridades que serão desenvolvidas no decorrer da semana, ressaltando que o XP baseia-se em requisitos atuais para o desenvolvimento de software e não em requisitos futuros, ao término da semana o resultado de todo o desenvolvimento é entregue ao cliente, podendo este colocar as funcionalidades desenvolvidas em prática.

2.5.2 Pequenas versões (small releases)

É comum à equipe de desenvolvimento liberar pequenas versões iterativas ao cliente, mesmo que o lançamento venha a ser muito pequeno ele ainda deve proporcionar valor para os negócios do cliente, como parte do planejamento de lançamento, o cliente trabalha em conjunto com a equipe de desenvolvedores para definir as 'user stories' e a ordem de desenvolvimento das mesmas.

O lançamento de releases pequenos tem como grande benefício o fato de poder usar cada versão como ponto de verificação onde medir a precisão da estimativa, durante a fase de planejamento os desenvolvedores costumam estimar as 'user stories' com base no seu rendimento esperado e dificuldade. O ciclo de pequenos lançamentos é viável, pois estamos construindo primeiro as funcionalidades mais importantes.

2.5.3 Metáfora

Utilizam-se metáforas no projeto para facilitar a comunicação entre os membros da equipe, torna-se interessante o desenvolvimento de uma linguagem que facilite a comunicação entre seus membros, e de fácil assimilação, ou seja, as metáforas são as

descrições de um software sem a utilização de termos técnicos, visando facilitar a comunicação entre o cliente e o desenvolvedor com a maior transparência possível para o cliente. Um exemplo de metáfora é quando a equipe está desenvolvendo um sistema de investimento no mercado financeiro, o cliente estará sempre usando termos como 'taxas de juros simples', 'composto', 'valor futuro de fluxo em caixa', 'capitalização', assim por diante. A equipe alinha a sua linguagem em torno da linguagem do cliente.

2.5.4 Design simples

No início do projeto o design deve ser o mais simples possível e ir melhorando de forma continua no decorrer do mesmo, no extreme programming existe uma definição bem clara sobre o significado de simplicidade:

- Não conter nenhum código duplicado;
- Conter o menor número possível de classes e métodos;
- Afirmar claramente a intenção do programador para o código;
- Que seja de fácil utilização;
- Que possa evoluir com o tempo.

2.5.5 Testes

No início do projeto os usuários definem regras para os testes de aceitação, tais testes serão utilizados como métricas e a partir destas são realizados os testes de forma unitária sobre o código produzido, através dos testes conseguimos a redução dos erros, aumentando a fidelidade do projeto.

2.5.6 Refatoração

Refatoração (melhoria constante do software), a cada nova funcionalidade a ser implantado o código estará sempre em sua forma mais simples, mesmo que seja necessário alterar quaisquer outras partes que já esteja funcionando, essa prática nem sempre é bem aceita, pois pode afetar o prazo e os custos.

2.5.7 Programação Pareada

Programação pareada ou em dupla, tem como pontos fortes a ampliação da comunicação da equipe de desenvolvimento, nivelar o conhecimento dentro da equipe, diminuição de erros corriqueiros, maior autonomia do código, correção de problemas complexos com melhor arquitetura.

2.5.8 Propriedade Coletiva do Código Fonte

O código fonte não possui dono, sendo assim qualquer membro da equipe de desenvolvedores pode altera-lo a qualquer momento, não sendo necessário solicitar permissão para isso, pois o código é de propriedade coletiva.

2.5.9 Cliente Ativo

Para um bom andamento do projeto é muito importante que o cliente esteja sempre disponível, seja para sanar quaisquer dúvidas sugerir mudanças e as prioridades que ele necessite.

2.5.10 Padrão de Codificação

O XP recomenda que ao início do projeto deve-se adotar uma padronização do código por parte da equipe de desenvolvedores, caso haja a necessidade de alterações e adaptações pelo decorrer do projeto visando uma maior agilidade no desenvolvimento deste; elas serão feitas. Todo trabalho desenvolve-se seguindo um padrão, porém é de vital importância que todos na equipe sigam esse mesmo padrão, sendo assim todos terão a mesma visão do código.

2.5.11 Integração Contínua

Os módulos de software são integrados várias vezes no decorrer dos dias e vários testes são realizados, o mesmo só é aprovado quando obtém 100% de êxito nos testes.

2.5.12 Semana de 40 horas

O XP define que um ritmo de trabalho sadio e que possa ser mantido sem prejudicar a saúde da equipe de desenvolvimento é de 40 horas semanais, sem a necessidade de fazer horas extras. Realizar o trabalho além do período estipulado pode ser considerado como normal, porém o fato de realizar horas extras por períodos superiores a uma semana é indício que há algo errado com o projeto, segundo Medeiros (2012): "Trabalhe a 100% durante as 40 horas e descanse a 100% no resto. Se algum deles não for feito com 100%, um afetará o outro."

Ou seja, é importante saber dosar o trabalho sem sobrecarregar a equipe de trabalho, é importante gerar condições favoráveis deixando a equipe de forma livre e descontraída, diminuído assim o desgaste físico e mental.

2.5.13 Stand Up Meeting

A stand up meeting é uma reunião breve que leva no máximo 20 minutos de duração, ela ocorre de forma diária, e seus participantes permanecem preferencialmente em pé. As reuniões em pé são mais rápidas, evita conversas que não façam parte do contexto e faz com que os participantes foquem diretamente o assunto.

2.5.14 Desenvolvimento guiado por testes

Ao contrário das metodologias tradicionais, no XP, para todo novo código implementado deverá ser realizado um teste, garantido desta forma o funcionamento da nova funcionalidade. Com embasamento nos testes de funcionalidades é que o desenvolvedor terá coragem de modificar uma parte do código que ainda não fora implementada.

No XP existem os testes de aceitação que são descritos pelo cliente e implementado pelos desenvolvedores, e os testes de unidade que tem um enfoque verificar se os resultados concebidos por cada classe estão corretos.

2.6 Ciclo de vida de um Projeto de Programação XP

Um projeto passa por algumas fases durante o seu ciclo de vida, dentro destas fases são executadas várias tarefas. Abaixo iremos abordar as principais fases de um projeto, obtendo uma ideia de como o projeto caminha ao logo de seu desenvolvimento.

2.6.1 Fase de Exploração

Nesta fase ocorre a análise de possíveis soluções e a viabilidade destas, no ponto de vista de requisitos, é interessante detalhá-los ao máximo conseguindo assim uma visão suficiente para o início do projeto. Conforme vai ocorrendo os avanços, posteriores aos de planejamento, as 'users stories' vão sendo priorizadas.

2.6.2 Fase de Planejamento

A fase de planejamento inicial é utilizada para que os clientes entrem em acordo sobre uma data de lançamento do primeiro release. Nessa fase o planejamento funciona da seguinte maneira: o cliente e os programadores em conjunto definem as 'user stories' que serão implementadas. Para cada ‘user stories' os programadores definem a velocidade de sua implementação baseada na dificuldade de desenvolvimento da mesma, normalmente as 'stories' com maior dificuldade são escolhidas para serem implementadas primeiro. O tempo para cada iteração é de uma a três semanas, já para cada release o tempo é de dois a quatro meses.

2.6.3 Fase de Iteração

É nesta fase que são escritos os testes funcionais de unidade, os desenvolvedores seguem determinadas atividades em cada iteração, a escrita dos casos de teste, refatoramento, codificação, realização de testes, integração. Conforme essas atividades vão sendo seguidas, o sistema vai sendo construído segundo os princípios, valores e práticas. Após o término do primeiro release, já iremos ter uma noção melhor das tecnologias e dos problemas envolvidos de forma que as próximas iterações poderão ser menores e nos próximos releases, podendo realizar estimativas mais precisas com base na experiência adquirida nas iterações passadas.

Após o final do primeiro release, dá-se o início da fase de produção, onde cada novo release depois de construído é posto para rodar em um ambiente que venha simular o ambiente de produção com intuito de verificar seu desempenho, comportamento, podendo ou não realizar testes adicionais de aceitação.

2.6.4 Fase de Manutenção

Nesta fase você está simultaneamente desenvolvendo novas funcionalidades, mantendo o sistema já existente rodando e reunindo novas pessoas ao projeto e melhorando o código existente. Nos projetos em XP, a introdução de novas tecnologias, novas ideias e refatoramento é bem aceita em qualquer fase do projeto, porém é importante ressaltar que a manutenção que será dada a um sistema deve ser realizada com a máxima cautela, pois quaisquer erros nas modificações podem acarretar no entrave do sistema ocasionado prejuízos ao cliente.

2.6.5 Fase de Morte

A fase de morte corresponde à conclusão de um projeto em extreme programming, há duas razões para o término de um projeto, a primeira seria quando o cliente encontra-se satisfeito com o projeto não visualizando nenhuma nova funcionalidade, já a segunda razão seria o projeto ter se tornado inviável economicamente, devido à dificuldade de realizar novas implementações a um valor relativamente baixo devido a um grande número de erros.

2.7 Papéis na Programação XP

Em uma equipe de desenvolvimento que utilize o XP, existem alguns papéis a serem desempenhados por um ou mais desenvolvedor, as quais buscaremos mostrar abaixo.

2.7.1 Treinador (Coach)

O treinador é a pessoa responsável pela execução técnica e por toda a evolução do processo, ele deve ser um bom comunicador e sempre manter a equipe motivada para

não perder o foco no desenvolvimento, também auxilia a equipe em tudo que for necessário. O treinador também é a pessoa que negocia com o cliente o escopo de toda nova iteração.

2.7.2 Rastreador (Tracker)

A função do rastreador é coletar as métricas que vão sendo desenvolvidas e compara-las com as métricas que foram estimas, realizando uma análise das eventuais divergências, também é responsável por avaliar a viabilidade do objetivo conforme as limitações de recursos e tempo.

2.7.3 Programador

No XP o programador ocupa um papel de destaque já que ele é responsável pela codificação do programa, pela organização e realização de testes e mantem o código fonte mais simples possível. Além disso, o programador é responsável pelas estimativas das 'user estories', e realiza modificações nestas estimativas quando necessário.

2.7.4 Cliente

O cliente é responsável por definir a ordem de prioridade de implementação para cada requisito, ele também seleciona e valida os requisitos do sistema, iteração por iteração, também é o único que pode eventualmente decidir sobre a redução do escopo da iteração objetivando a entrega dentro do prazo.

3 Comparativo entre as metodologias ágeis Scrum e XP

Abaixo segue relacionado o comparativo entre o Scrum e o XP, onde podemos diferenciar algumas vantagens de uma metodologia em relação à outra.

Métodos / Fases	Scrum	Extreme Programming (XP)
Requisitos Iniciais	Os requisitos do sistema são listados gerando o Product Backlog	User Storeis escrita pelo cliente
Atribuir Requisitos ao Sistema	Os requisitos definidos no Product Backlog são alocados as Sprints, durante a reunião de planejamento da Sprint.	Definição das User Stories que serão desenvolvidas a cada iteração
Levantamento de Requisitos	Proprietário do produto mantém Product Backlog. Não há nenhuma fase de elaboração separada.	Cliente escreve/coleta histórias. O resultado é uma pilha de cartões de história. Não há nenhuma fase de elaboração separada, embora o cliente deva elaborar quanto for necessário no início da iteração (Timebox).
Projetar Arquitetura do Sistema	Projeto geral baseado no Product Backlog	Paralelo ao desenvolvimento da User Stories
Desenvolver Incremento do Sistema	Implementação dos requisitos contemplados no Sprint Backlog. Dayle Meetings de 15 a 30 minutos.	Implementação das User Stories que fazem parte da iteração corrente por dupla de programadores
Validar Incremento	Acrescente ao final da Sprint	Programadores: testes de unidade Cliente: testes de aceitação Ambos os testes são escritos antes da codificação e executados após
Integrar Incremento	Acontece ao final de cada Sprint	Código é integrado à medida que vai sendo desenvolvido
Validar Sistema	O sistema é validado no último dia de cada Sprint (Revisão da Sprint)	Sistema é validado pelo Cliente
Entrega Final	Todos os itens do Product Backlog desenvolvidos	Cliente satisfeito
Número de equipes	1 a 4 ou mais	1 equipe por projeto
Tamanho da Equipe	5 a 9 pessoas	3 a 16 pessoas
Membros da Equipe/Funções	Scrum Master, Time Scrum (programadores)	Cliente, programador, testador, treinador.
Funções no projeto	Gerente de projetos / Scrum master, Product Owner	Cliente, Programador

Considerações finais

Dentre as metodologias ágeis as mais conhecidas e utilizadas são respectivamente o Scrum e o XP (Extreme Programming), ambas têm suas semelhanças

e respectivamente suas diferenças, porém por serem técnicas de desenvolvimento ágil de software, elas possuem as mesmas características que são: tempo relativamente curto e consiste de várias iterações.

As pessoas muitas vezes confundem o Scrum com o XP, porém o Scrum é um subconjunto de técnicas de desenvolvimento de software ágil. Sendo um processo iterativo para desenvolver ou gerenciar certos projetos e consequentemente irá produzir produtos que possam ser apresentados ao cliente no final de cada Sprint. Ao contrário do XP, o Scrum enfoca o lado da gestão de um projeto, porém não define regras como o XP.

No Scrum as iterações são chamadas de Sprint, sendo esta que irá lidar com certos recursos definidos pelo cliente. Uma Sprint dura trinta dias podendo ser reduzida há uma semana dependendo do tamanho e complexidade do projeto. Durante uma Sprint a equipe de desenvolvimento (Scrum Team) realizará reuniões diárias (Daily Meetings), apresentando o que foi desenvolvido, quais as dificuldades apresentadas e impedimentos e o que será desenvolvido.

O XP consiste de várias práticas e valores, sendo elas: comunicação, simplicidade, feedback rápido e coragem. Os integrantes da equipe em um projeto XP são formados, respectivamente, desenvolvedores e representantes do cliente. O XP também enfoca o design simples e a solução para cumprir exigências do cliente. Já no processo de desenvolvimento, o feedback por parte do cliente é de vital importância para o bom andamento do projeto, assim como a realização de testes de funcionalidade. O XP enfatiza a programação em pares, com o intuito de desenvolver o código somente para as exigências atuais e a refatoração do código caso faça-se necessário no futuro.

O intuito deste trabalho inicialmente era propor uma metodologia hibrida entre o Scrum e o XP, porém o XP exige times altamente disciplinados para execução e comprimento de suas métricas, desta forma este mostrou também algumas dificuldades nesse sentido, sendo elas:

- No XP o código é de propriedade coletiva, o que muitas pessoas acreditam ser uma vantagem em relação ao Scrum, pois a qualquer momento você pode modificar o código sem precisar iniciar uma nova Sprint no caso do Scrum, porém o fato de qualquer pessoa poder alterar o código a qualquer momento sem qualquer controle é muito capaz de comprometer todo o trabalho já desenvolvido.

- O XP é complexo, porém suas práticas e valores simples, mas é preciso muita concentração e esforço para seguir um verdadeiro projeto em XP.
- O XP se baseia em muitas regras e métricas, sendo um processo minimalista, baseando-se em um conjunto de práticas para compensar a falta de severidade em outras áreas. Caso algo de errado aconteça, todo o projeto corre o risco de fracassar.

Dessa forma conclui-se que utilizar o XP propriamente dito ao pé da letra junto ao Scrum pode representar uma grande dificuldade, o que seria mais indicado nesse processo é a equipe de desenvolvimento iniciar o projeto utilizando o Scrum e conforme o grupo for adquirindo maturidade suficiente ir adaptando a sua própria versão do XP.

Referências

BAIRD, S. **Extreme Programming Practices in Action.** 2002. Disponível em: <http://www.informit.com/articles/article.aspx?p=30187&seqNum=3>. Acesso em: 13 out. 2012.

BECK, K. et al. **Manifesto para desenvolvimento Ágil de Software**. Disponível em: <http://www.agilemanifesto.org>. Acesso em: 19 ago. 2012.

_____. **Princípios por trás do Manifesto Ágil**. Disponível em: <http://agilemanifesto.org/principles.html>. Acesso em: 19 ago. 2012.

COHN, M**. User stories applied for Agile Software Developmente.** 2004. Disponível em: <http://www.userstories.com/book>. Acesso em: 29 ago. 2012

GAMBA, L. M. et al. **Aplicações de métricas de Software com Scrum**. Disponível em: <http://periodicos.unesc.net/index.php/sulcomp/article/view/242/247>. Acesso em: 14 ago. 2012

LEITE, C. J. **O modelo cascata.** mar. 2007. Disponível em: <http://engenhariadesoftware.blogspot.com.br/2007/03/o-modelo-cascata.html>. Acesso em: 4 set. 2012.

MEDEIROS,P. M. **Extreme Programming - Conceitos e Práticas**. Disponível em: <http://www.devmedia.com.br/extreme-programming-conceitos-e-práticas/1498>. Acesso em: 15 de jun. 2012.

PHEAMES, J. **TDD:** is there really any debate any longer? Disponível em: <http://blog.pluralsight.com/2012/10/15/tdd-is-there-really-any-debate-any-longer/>. Acesso em: 15 out. 2012.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum**: um guia definitivo para o Scrum: as regras do jogo. Disponível em:

</Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20Portuguese%20BR.pdf>. Acesso em: 17 ago. 2012.

STEFFEN, B. J. **O que são essas tais de metodologias Ágeis?** Disponível em: <https://www.ibm.com/developerworks/mydeveloperworks/blogs/rationalbrasil/entry/mas_o_que_s_c3_a3o_essas_tais_de_metodologias__c3_a1geis?lang=en>. Acesso em: 2 set. 2012

TELES, M. V. **Um estudo de caso da adoção das práticas e valores do Extreme Programming,** Disponível em: < http://improveit.com.br/xp/dissertacaoXP.pdf> Acesso em: 14 out. 2012.

VIZDOS, M. **The Classic Story of the Pig and Chicken.** Disponível em: <http://www.implementingscrum.com/2006/09/11/the-classic-story-of-the-pig-and-chicken/>. Acesso em: 30 ago. 2012.