UNINOVE

Metodologias Ágeis: Extreme Programming e Scrum

APRESENTAR OS PRINCÍPIOS E FUNDAMENTOS DAS METODOLOGIAS ÁGEIS XP (EXTREME PROGRAMMING) E SCRUM.

Introdução

Às vezes, ao ter que desenvolver um software, o cliente pode necessitar de rapidez na entrega. Nem sempre ao aplicar os modelos de processos mais tradicionais isso é possível.
Pensando nesse tipo de necessidade, por volta de 2001, alguns desenvolvedores propuseram uma nova abordagem para o processo de desenvolvimento de software, denominada metodologia ágil.
Nessa nova metodologia, o projeto é visto como um conjunto de pequenas tarefas ao invés de um processo completo, em que se procura aumentar a leveza e a rapidez no desenvolvimento do software.
Vários modelos de processo foram propostos seguindo os princípios das metodologias ágeis, como XP e *Scrum*. Neles, as atividades são estabelecidas seguindo alguns princípios, dentre os quais podemos citar:

- Indivíduos e iterações acima de processos e ferramentas.
- Software funcionando acima de documentação abrangente.
- Colaboração com o cliente acima de negociação de contratos.
- Responder a mudanças acima de seguir um plano.

XP (Extreme Programming)

XP é uma metodologia ágil de desenvolvimento de software. O seu principal objetivo é criar sistemas de qualidade através de um conjunto de valores, princípios e práticas que diferem bastante das tradicionais.
A princípio, é indicada para equipes pequenas e médias, desenvolvendo software com requisitos vagos ou que mudam frequentemente.
Seu foco é a codificação (de preferência aos pares), com o código sendo propriedade coletiva. Os valores em que se sustenta são comunicação simples, porém eficientes; simplicidade no design, algoritmo e tecnologias utilizadas; feedback em relação à qualidade do código e ao andamento do projeto e coragem para aplicar mudanças que venham a surgir durante o desenvolvimento.

Ela enfatiza o trabalho em equipe (gerentes, clientes e desenvolvedores são todos parceiros em um trabalho colaborativo) e introduz um ambiente de trabalho simples e efetivo, permitindo que as equipes se tornem altamente produtivas.

A equipe se auto-organiza em torno de um problema, visando resolvê-lo da maneira mais eficiente possível.

Talvez o maior problema no desenvolvimento de software seja incorporar as solicitações de mudanças nos requisitos durante o desenvolvimento. Os processos ágeis reconhecem e acomodam essa realidade das mudanças.

As iterações são curtas. As mudanças devem ser incrementais e feitas aos poucos.

Os planejamentos de release e das iterações são feitos com base na estória dos usuários e conta com a colaboração de toda a equipe de desenvolvimento, inclusive o cliente. Tais estórias descrevem cenários com situações de utilização que os envolvidos gostariam que o sistema viesse a oferecer.

A princípio, devem ser escritas pelos próprios usuários. São análogas aos casos de uso da UML, mas não são a mesma coisa. A diferença é que elas não se limitam a descrever o processo de interação do usuário com o sistema.

No XP, elas substituem longos documentos de requisitos dos métodos tradicionais. São a base para a criação de estimativas de tempo que serão utilizadas nas reuniões de planejamento de entregas (*releases*). Cada estória deve levar de 2 a 3 semanas para ser implementada em uma iteração individual. Se levar mais do que isso, é necessário que seja dividida em duas ou mais.

São, também, utilizadas como base para a elaboração dos testes de aceitação. Um ou mais testes de aceitação automatizados devem ser criados para verificar se o programa implementa a estória corretamente.

Nota-se, portanto, que o envolvimento do usuário como autor das estórias é de suma importância para a elaboração do plano de entregas e dos testes de aceitação em cada iteração de desenvolvimento.

No final de um *release*, é feita uma determinação rápida do escopo do próximo, através da combinação de estimativas e prioridades do negócio.

Um *release* consiste de várias iterações e, em cada iteração, várias estórias são implementadas.

Os programadores estimam cada estória e dizem quantas eles podem implementar no final do *release*.

Baseados nesses dados, os clientes escolhem as principais estórias, que serão implementadas (de preferência, aquelas que agregam maior valor para o negócio).

No XP, faz-se uso de metáforas, com a intenção de oferecer uma visão geral do sistema de uma forma simples e que possa ser compartilhada por clientes e programadores.

Utiliza-se, também, a refatoração, que se baseia em constantes melhorias no projeto do software para aumentar sua capacidade de se adaptar a mudanças.

A semana de trabalho dos envolvidos é, preferencialmente, de 40 horas, já que na visão da XP, o cansaço e a insatisfação de trabalhar horas extras pode levar a uma queda da qualidade do código.

Esta metodologia prega a utilização de padrões de codificação, ou seja, que todos os programadores programem da mesma forma, facilitando o entendimento do código e as alterações realizadas. Fortificando o princípio da propriedade coletiva do código.

Basicamente, um projeto XP passa pelas fases de exploração, planejamento inicial, iterações do release, produção, manutenção e morte.

SCRUM

Dentro dos princípios das metodologias ágeis, o *Scrum* objetiva, de forma iterativa, entregar ao cliente, incrementos de produto de alto valor.

Portanto, os projetos são desenvolvidos através de uma série de iterações, cada uma com duração de uma semana a um mês, denominadas incrementos (*sprints*).

Assim como o XP, *Scrum* é adequado, principalmente, para projetos com requisitos variáveis e que exigem rapidez de entrega.

O trabalho a ser realizado é registrado nas pendências do produto (*Product Backlog*): uma lista dos requisitos do produto.

No início de cada incremento (sprint) é realizada uma reunião de planejamento de incremento (Sprint Planning Meeting), na qual o dono do produto (Product Owner) informa os requisitos do produto (Product Backlog) que ele quer que sejam concluídos, e a equipe Scrum (Scrum Team) seleciona as tarefas que são possíveis de se realizar durante o próximo incremento.

Essas tarefas são, então, movidas do Product Backlog para o Backlog do Sprint. A execução de cada sprint deve terminar dentro do "quadro de tempo" previsto. Se não for possível implementar todos os requisitos contidos no backlog daquele sprint, eles são colocados de volta no backlog do produto.

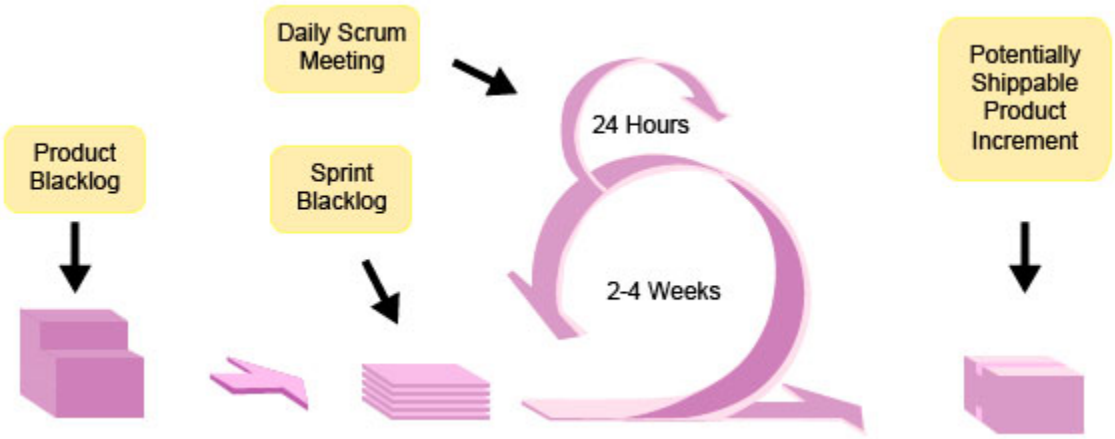
Em cada sprint há a realização de reuniões diárias, denominadas Scrum Diário (Daily Scrum, Daily Meeting ou ainda Stand up Meeting). Essas reuniões são importantes, pois ajudam a equipe a manter o foco. É onde cada participante descreve o que foi feito, o que deve ainda ser realizado e informa sobre possíveis dificuldades que o impedem de avançar.

No final de cada sprint, a equipe apresenta a funcionalidade concluída, na denominada reunião de revisão do incremento (Sprint Review Meeting).

Os participantes característicos do Scrum são:

- O proprietário do produto (Product Owner), que representa os stakeholders e o negócio.
- O ScrumMaster, que tem a função de manter os processos (pode-se fazer uma alusão a um gerente de projeto).
- A equipe (Team): grupo pequeno de pessoas que executam as etapas de desenvolvimento

A Figura 1 ilustra de uma forma esquemática o modelo *Scrum.*



Legenda: FIGURA 1 ? DIAGRAMA ESQUEMáTICO DO SCRUM. FONTE: PRESSMAN, 2010.

Agora que você já estudou esta aula, resolva os exercícios e verifique seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.
EXERCÍCIO (https://ead.uninove.br/ead/disciplinas/impressos/_g/pdsoft80_100/a15ex01_pdsoft80_100.pdf)

REFERÊNCIA

PRESSMAN, R. S. *Engenharia de software*. 7. ed. São Paulo: McGraw-Hill, 2010.

SOMMERVILLE, Ian. *Engenharia de software*.São Paulo: Addison-Wesley, 2007.