

Gestão Ágil de Projetos

Professor – Ramon Trigo



Processo Ágil de Desenvolvimento

Na literatura, existem diversos modelos de processo de software, tais como: cascata, prototipação, incremental, iterativo, espiral, entre outros.

O SCRUM é um *framework* de gerenciamento de projetos ágeis, tem como base equipes pequenas e multidisciplinares, os *feedbacks* constantes e a colaboração dos envolvidos.

O modelo XP tem muita semelhança com o SCRUM em termos de valores e modelo de desenvolvimento de projetos, entretanto, ele foca mais em práticas de engenharia.

O modelo kanban, por sua vez, visa a aumentar a produtividade e otimizar a gestão do trabalho, utilizando cartões em um quadro para indicar e acompanhar, de maneira visual, prática e utilizando poucos recursos, o andamento dos fluxos de produção nas empresas.

Manifesto Ágil

Nós vivemos em um mundo que está em constante mudança e essas mudanças acontecem em uma alta velocidade, às vezes é difícil acompanhá-las.

A chave para isso é a adaptação. Quanto mais rapidamente as organizações e as pessoas conseguirem se adaptar aos novos desafios, mais rápido aparecerão os resultados e isso se torna um grande diferencial no mercado o qual essas organizações estão inseridas.

As metodologias conhecidas como ágeis surgiram de um trabalho colaborativo de 17 profissionais que se reuniram em busca de novas alternativas para desenvolvimento de produtos e serviços de forma mais ágil e eficiente.

Manifesto Ágil

Originalmente, o foco era o desenvolvimento de sistemas, mas atualmente este é utilizado em outros domínios.

O objetivo dessas metodologias era contornar as limitações dos modelos de processo de software existentes.

Para isso, o movimento criado, denominado de manifesto ágil, reúne um conjunto de 12 princípios que devem ser seguidos, sendo:

Manifesto Ágil

1. **Satisfação do cliente**: para satisfazer o cliente, serão realizadas entregas adiantadas e contínuas de software de valor.
2. **Mudanças no software**: deve-se aceitar que durante o projeto podem acontecer mudanças de requisitos e os processos ágeis devem se adequar às mudanças para que o cliente possa ter vantagens competitivas.
3. **Entregas curtas**: desenvolver versões funcionais para serem disponibilizadas ao cliente em prazos curtos, no prazo de semanas ou no máximo meses.
4. **Trabalhar em conjunto**: os *stakeholders* devem trabalhar em conjunto e diariamente durante todo o ciclo de vida do projeto.

Manifesto Ágil

5. **Motivação constante**: é importante que os integrantes da equipe do projeto tenham um ambiente e suporte necessários para realizarem o seu trabalho e o principal é depositar a confiança de que eles irão fazer o seu trabalho, gerando indivíduos motivados.
6. **Conversa aberta com a equipe**: é o método mais eficiente para transmitir informações para um time de desenvolvimento.
7. **Software funcional**: independente da etapa em que está sendo executado, o software sempre deve estar em funcionamento, é a medida primária de progresso e evolução.

Manifesto Ágil

8. **Ritmo constante**: a agilidade do desenvolvimento do projeto está ligada à capacidade de produção da equipe, sendo que horas extras e esforços adicionais não são recomendados.

A sustentabilidade das ações e das entregas é mantida através de um número de horas de trabalho adequado, que permite à equipe, o descanso devido para uma nova fase de desenvolvimento.

9. **Atenção contínua**: a gestão de projetos, baseada nos métodos ágeis, é incremental, ou seja, uma vez atingida a excelência técnica, outros fatores, como design e adequações, são desenvolvidos a cada nova iteração sem impactar na agilidade do desenvolvimento da solução.

10. **Simplicidade**: reduzir a carga de trabalho sem perder a qualidade da solução

Manifesto Ágil

11. **Times autogerenciáveis:** o comprometimento do time de projetos é fundamental para uma metodologia ágil, por isso o trabalho executado atinge um nível de excelência acima do normal. As pessoas sabem o que precisam fazer e assim o fazem, sem perda de tempo ou de recursos.

12. **Avaliação da equipe em tempos regulares:** em tempos regulares, o time de projetos se reúne para avaliar o seu desempenho e desenvolver novas maneiras de se tornar mais efetivo, contribuindo para que se torne ainda mais ágil e eficaz na execução de suas atribuições.

Extreme Programming

Extreme Programming (XP) é talvez o mais conhecido e mais utilizado dos métodos ágeis.

O nome foi cunhado por Beck (2000), pois a abordagem foi desenvolvida para impulsionar práticas reconhecidamente boas, como o desenvolvimento iterativo, a níveis 'extremos'.

Por exemplo, em XP, várias novas versões de um sistema podem ser desenvolvidas, integradas e testadas em um único dia por programadores diferentes.

Manifesto Ágil - Valores

Indivíduos e interações

Software que funciona

Colaboração do cliente

Resposta à mudanças

ao invés de

Processos e ferramentas

Documentação abrangente

Negociação de contrato

Seguir um plano

Extreme Programming

Extreme Programming é uma metodologia ágil que tem como valores principais comunicação, simplicidade, feedback, coragem e respeito.

É uma excelente abordagem para equipes pequenas e que tem constantes mudanças de escopo (alvo) do projeto.

Extreme Programming

Características:

- Método de desenvolvimento de software;
- Conjunto de valores, princípios e práticas;
- Ciclo de desenvolvimento curtos e releases frequentes;
- Permite respostas rápidas a mudanças de requisitos em qualquer etapa do desenvolvimento;
- Enfatiza o trabalho em equipe que se auto-organiza em que todos são iguais num trabalho colaborativo.

Cliente Presente

- O cliente deve participar ativamente do processo de desenvolvimento. Tudo precisa da comunicação com o cliente. Ele deve receber o melhor resultado possível a cada semana, ver o progresso no sistema, ser informado de mudanças de planos, etc.
- Escute, para que saiba qual é o problema a ser resolvido.

Planejamento

- O desenvolvimento utilizando o XP é feito em iterações. Uma iteração é um período curto de tempo (1 ou 2 semanas) onde a equipe desenvolve um conjunto de funcionalidades.
- Sendo assim, no início da semana, desenvolvedores e clientes se reúnem para priorizar as funcionalidades. Essa reunião chama-se jogo de planejamento e nela já devem estar criadas as estórias.

Planejamento

- .
- Se for muito grande, ela deve ser dividida em tarefas com duração máxima **estória** de alguns dias. **Essas estórias devem ser escritas pelo cliente,** pois assim ele consegue pensar melhor em cada funcionalidade.
- O planejamento é importante para que você sempre faça a coisa mais importante a ser feita.

Stand Up Meeting

- São reuniões feitas em pé e de curta duração – mas muito produtiva, para que o time se mantenha alinhado, para saber o que cada um está fazendo exatamente, em que ponto está o projeto e se alguém está tendo problemas para executar suas tarefas.
- Ainda que apareça algum problema, essa reunião não tem o propósito de pensar em soluções.

Programação em Par

- É uma programação em par (dupla) em um único computador. Como é apenas um computador, o software sempre é revisto por duas pessoas diminuindo assim a possibilidade de falhas.
- Busca-se sempre a evolução da equipe melhorando a qualidade do código fonte.
- Ela é uma das práticas primordiais do XP, pois dois programadores fazendo o trabalho juntos acaba agregando muito para o trabalho em equipe

Testes Constantes

- É utilizado o Desenvolvimento Orientado a Testes (Test Driven Development), o conhecido TDD. Primeiro crie os testes unitários e depois crie o código para que o teste funcione, essa abordagem é complexa no início, mas os testes unitários são essenciais para que a qualidade do projeto seja mantida.

Código Coletivo

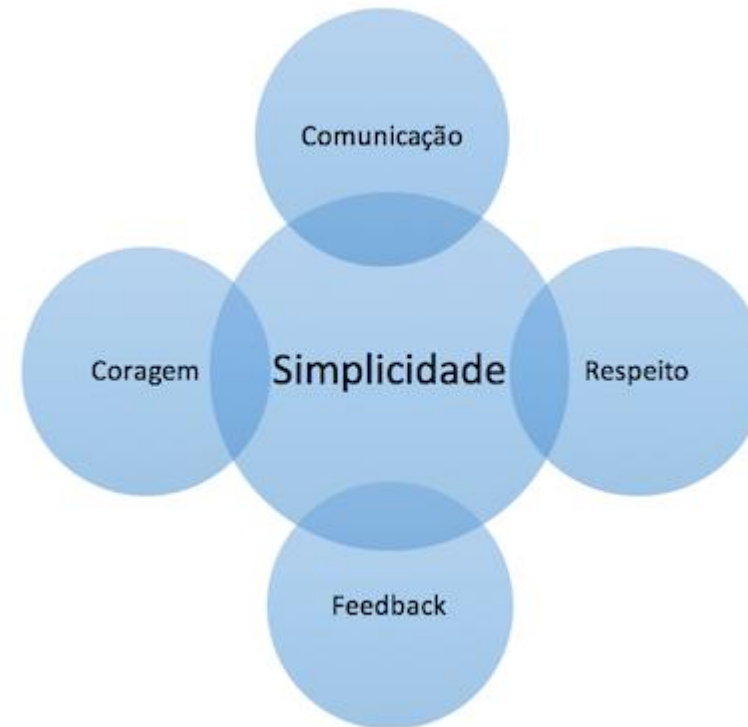
- Diz que o código fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo.
- O objetivo é fazer a equipe conhecer todas as partes do sistema.

Release Curtos

- As liberações de pequenas versões funcionais do projeto auxiliam muito no processo de aceitação por parte do cliente que já pode testar uma parte do sistema.
- As versões chegam ainda ser menores que as produzidas em outras metodologias incrementais, como o RUP.
- Os releases são pequenos pedaços do produto que são entregues ao cliente antes do tempo, assim o cliente não precisa esperar o produto todo ficar pronto para ver.

Características

Simplicidade;
Comunicação;
Feedbacks;
Coragem;
Respeito.



Simplicidade

- Não tente prever o futuro!
- Comece por soluções simples e que funcionem.
- Melhorias são adicionadas depois

Comunicação

- Todos são parte do time, inclusive o cliente!
- Contato diário e face a face.
- Pair programming.

Feed Back's

- Testes:
 - Unitários;
 - Integração.
- Cliente:
 - Testes funcionais.
- Membros do time:
 - Quando um novo requisito chega o time estima quanto tempo será necessário.

Coragem

- Não deixe para amanhã o que pode ser feito agora!
- Não ficou bom?! Refaça o quanto necessário!
- Se não precisa, jogue fora!

Respeito

- Ninguém modifica o código no repositório sem testar antes;
- Todos buscam a melhor solução, o melhor design para poupar retrabalhos futuros;
- Todos os valores anteriores são respeitados;
- Pessoas são valorizadas e ninguém é ignorado.

Práticas

- **Process** – On-site customer (**cliente no local**), testing (testes), small releases (**versões pequenas**), planning game (planejamento do jogo)
- **Programming** – Simple design (**projeto simples**), testing (testes), refactoring(reconstrução), coding standards(**código padrão**) ;
- **Team** – Collective ownership (**código coletivo**), continuous integration (**integração continua**), pair programming (**programação em par**), small releases (versões pequenas) ;

Atividades

1. Qual é o princípio fundamental por trás do método ágil XP (Extreme Programming)?
2. Como o XP aborda a comunicação e a colaboração entre os membros da equipe de desenvolvimento?
3. Quais são os principais papéis e responsabilidades dentro de uma equipe de XP?
4. Como o XP lida com a questão da qualidade do código e a prática de desenvolvimento orientado a testes?
5. Quais são as práticas de programação que o XP enfatiza para garantir a entrega contínua de valor ao cliente?
6. Como o XP aborda a questão da adaptação a mudanças nos requisitos do cliente durante o desenvolvimento do software?
7. Qual é a importância da integração contínua no contexto do XP?
8. Como o XP lida com o processo de planejamento e estimativa de projetos?
9. Quais são os desafios comuns enfrentados pelas equipes que adotam o método ágil XP?
10. Como a cultura organizacional e a gestão de stakeholders podem influenciar a eficácia da implementação do XP em uma equipe ou empresa?