Processos Ágeis de Desenvolvimento de Software

Yuri Pereira

ycssp@cin.ufpe.br

UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Contexto

- Processos ágeis surgiram como alternativa aos processos tradicionais...
- ... que apresentam restrições principalmente às pequenas e médias fábricas de software...
- ... e tendem a planejar grande parte do desenvolvimento de software por um longo período, antes de iniciar a implementação
- Pouca susceptibilidade à mudanças
- Implementação de funcionalidades que não agregarão valor ao cliente, enquanto outras mais importantes não foram implementadas

Processos ágeis

- Desburocratização do processo de desenvolvimento
- Adaptação e fortalecimento com as mudanças
- Versões de software executável em curto espaço de tempo
- Funcionalidades que agregam maior valor ao negócio são priorizadas

Processos ágeis

- Não “documentar por documentar”, somente quando for estritamente necessário
- A documentação se restringe às estórias dos usuários e descrições do funcionamento do sistema
- Foco nas pessoas, não nos processos

O Manifesto Ágil

- Em 2001, 17 especialistas reunem-se para propor um conjunto de princípios e valores para agilizar o desenvolvimento dos seus sistemas, tendo como base suas experiências em programação
- Haviam concluído que os processos de desenvolvimento estavam se tornando cada vez mais burocráticos, dificultando a visibilidade e o entedimento das equipes de construção de software
- Definição de um novo enfoque, calcado na agilidade, flexibilidade e nas habilidades de comunicação

O Manifesto Ágil

Nós estamos descobrindo melhores maneiras de desenvovler software, fazendo e ajudando outros a fazê-lo. Através deste trabalho, passamos a valorizar:

Indivíduos e interação entre eles, mais que processos e ferramentas

Software em funcionamento, mais do que documentação abrangente

Colaboração com o cliente, mais do que negociação de contratos

Responder a mudanças, mais do que seguir um plano

O Manifesto Ágil - Princípios

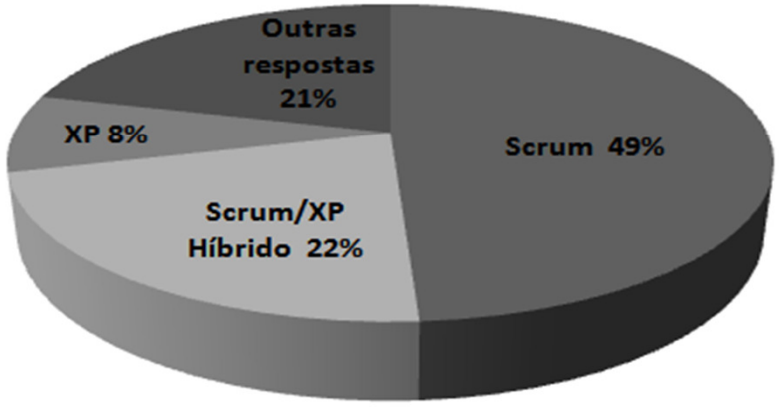
- A maior prioridade é satisfazer o cliente através de entregas antecipadas e contínuas de software de valor
- Processos ágeis vêem as mudanças como uma oportunidade de tirar vantagem
- Entrega freqüente de software funcionando em intervalos curtos de tempo
- Pessoas das áreas de software e de negócios trabalhando juntas diariamente
- Indivíduos motivados. Dê-los o ambiente e o suporte necessário e acredite neles

O Manifesto Ágil - Princípios

- O método mais eficiente e efetivo de repassar a informação é a conversa face a face
- Software funcionando é a primeira medida do progresso
- Processos ágeis promovem desenvolvimento sustentável
- Atenção contínua à excelência técnica e um bom projeto melhoram a agilidade
- Simplicidade é essencial
- As melhores arquiteturas, requisitos e projetos emergem de times auto-organizáveis
- Reflexão sobre como se tornar mais efetivo em intervalos regulares, ajuste de comportamento de acordo com a reflexão

Processos Ágeis

- São todos os processos que seguem os princípios do manifesto ágil



Scrum	49,1%
Scrum/XP Híbrido	22,3%
Extreme Programming (XP)	8,0%
Customizado/Híbrido	5,3%
Agile Unified Process (AgileUP)	2,2%
Feature-Driven Development (FDD)	2,1%
Lean Development	1,9%
Dynamic Systems Development Method (DSDM)	1,4%
OpenUP	0,6%
Agile Modeling	0,6%
Crystal	0,5%
Customizado/Híbrido	5,3%
Outras	2,2%
Não sabem	3,7%

Extreme Programming (XP)

- Criado em 1999 por Kent Beck e Ward Cunningham, quando do lançamento do livro *Extreme Programming Explained*
- Voltado ao desenvolvimento de produtos com requisitos não totalmente definidos e em constante mudança
- Apoiado em valores, princípios e práticas

Valores de XP

- **Comunicação: XP inova e ousa ao priorizar a comunicação pessoal e oral ao invés da escrita**
- **Simplicidade: XP recomenda manter o sistema simples, de forma a não gerar mais trabalho desnecessário.**
- **Feedback: Ao longo do desenvolvimento, é muito importante que exista *feedback* dentro do time de desenvolvimento e com o cliente e/ou demais envolvidos**
- **Coragem: O time pode ter coragem de refatorar, pois sabe que os testes irão detectar erros imediatamente, o cliente pode decidir com mais coragem, quando avalia o software funcional após cada release, sabendo que pode priorizar as funcionalidades que lhe são mais importantes**

Valores de XP

- **<u>Respeito:</u> valorizar a relação entre membros da equipe e, também, a de cada membro com o projeto e sua instituição.**

Princípios do XP

- **Humanidade: reconhece que são pessoas que desenvolvem software**
- **Economia: busca garantir valor para o negócio, refletir sobre o valor do dinheiro e como empregá-lo melhor ao longo do tempo**
- **Benefício mútuo: as práticas são estruturadas de modo a serem mutuamente benéficas para todos os envolvidos em um projeto se software**
- **Auto-semelhança: sugere que, quando equipes XP encontrarem soluções que funcionem em um contexto, também procurem adotá-las em outros, mesmo que em escalas diferentes**

Princípios do XP

- **Melhoria: indica que não devemos esperar a perfeição, mas sim fazer o melhor que pudermos hoje, para poder fazer o melhor amanhã**
- **Diversidade: lembra que um time com pessoas diferentes apresenta mais habilidades, conhecimentos e oportunidades**
- **Reflexão: implica em pensar como e por que trabalhamos**
- **Fluxo: Determina que exista uma corrente contínua de atividades e que o processo deve explicitá-la**

Princípios do XP

- Oportunidade: leva a encarar problemas como oportunidades para mudança
- Redundância: aumenta as chances de sucesso, promovendo várias oportunidades de fazer a coisa certa
- Falha: indica que pode ser bom falhar, desde que se aprenda com a experiência. Quando uma equipe não sabe para onde ir, arriscar-se a falhar pode ser o caminho mais curto para obter o sucesso
- Qualidade: Este princípio diz que quanto maior a qualidade, mas fácil será realizar o trabalho

Princípios do XP

- **Passos pequenos: garante que iremos fazer sempre o caminho mais curto na direção correta, pois a execução de tarefas complexas em passos pequenos diminui o risco**
- **Aceitação de responsabilidade: evidencia que só o próprio indivíduo pode se responsabilizar por suas ações**

Práticas do XP

- Sentar juntos: deixa explícita a necessidade de se ter um espaço onde toda a equipe possa trabalhar junta, valorizando a comunicação e possibilitando que as pessoas possam beneficiar-se de todos os seus sentidos ao conversar
- Time completo: a equipe precisa de pessoas com todas as habilidades e perspectivas necessárias para o sucesso do projeto
- Espaço de trabalho informativo: Essa prática diz que qualquer observador interessado deve ser capaz de olhar para o espaço de trabalho e ter ideia do andamento do projeto em pouco tempo

Práticas do XP

- **Estórias: São textos simples que expressam necessidades do cliente e que são estimadas, o quanto antes, pelos programadores**
- **Ciclo semanal: explicita as iterações que fazem parte de um *release.***
- **Ciclo de estação: explicita o planejamento de um release**
- **Folga: reconhece que por melhor que tenha sido o planejamento, o mesmo sempre falha**
- ***Build* veloz: exige que o sistema seja compilado por completo e todos os testes sejam executados, de maneira automática, em no máximo 10 minutos**

Práticas do XP

- **Design incremental: implica em investimento diário no *design* da aplicação**
- **Programação em pares: todo o código deve ser produzido por pares de programadores, cada par trabalhando na mesma máquina**
- **Integração contínua: o código é integrado, o *build* do *software* é gerado e os testes são executados várias vezes ao dia.**

Práticas Corolário do XP

- Implantação incremental
- Continuidade da equipe: propõe que equipes eficientes continuem trabalhando juntas
- Redução da equipe: com o passar do tempo, a necessidade de uma equipe grande pode diminuir, principalmente se o sistema entra em um ciclo de manutenção
- Análise da causa inicial: sempre que encontrar um defeito, elimine o defeito e sua causa, para que o mesmo tipo de erro não ocorra novamente.
- Código e testes: explicita que só o código fonte e os testes automatizados devem ser artefatos permanentes gerados por uma equipe XP

Práticas Corolário do XP

- **Repositório único de código: todo o código deve estar contido em um único repositório que não deve ter *branches* permanentes**
- **Implantação diária: é a evolução da prática de *releases* pequenos**
- **Contrato de escopo negociável: determina como devem ser feitos contratos em um projeto que adota XP, fixando o tempo, custos e a qualidade, mas mantendo o escopo negociável**
- **Pague pelo uso: também aborda o lado de negócios de um projeto XP, sugerindo que o cliente pague por toda vez que for usar o sistema**

Papéis no XP

- Programadores
- Arquitetos
- Analista de Testes
- Analistas de Negócios
- Projetistas de Interação
- Gerente de Projetos
- Gerente do Produto
- Usuários

Ciclo de vida no XP

- **Fase de exploração: Nela, investigações são feitas e é verificada a viabilidade de possíveis conclusões/soluções serem implementadas.**
- **Fase de planejamento**
- **Fase das iterações do *release*: de posse do planejamento da iteração, o plano de iteração é posto em desenvolvimento, momento em que os programadores seguem um fluxo de atividades**
- **Fase de aprovação**
- **Manutenção e Morte:**

SCRUM

- Criado em 1996 por Ken Schwaber e Jeff Sutherland
- Destaca-se dos demais processos ágeis pela maior ênfase dada ao gerenciamento do projeto
- Reúne atividades de monitoramento e *feedback*, em geral, reuniões rápidas e diárias com toda a equipe...
- ... visando a identificação e correção de quaisquer deficiências e/ou impedimentos no processo de desenvolvimento

Características do SCRUM

- Focado em pessoas e em ambientes onde há requisitos voláteis
- Encontrar uma forma de trabalho dos membros da equipe para produzir o software de forma flexível num ambiente em constante mudança
- equipes pequenas e multidisciplinares (no máximo 7 pessoas), requisitos instáveis ou desconhecidos e iterações curtas
- Cada ciclo do *Scrum* é denominado *Sprint*, possuindo intervalos de tempo reduzido de 15 a 30 dias
- No *Scrum* o cliente toma parte da equipe de desenvolvimento, através da figura do *Product Owner*

Papéis no SCRUM

- ***Scrum Master*** **(SM)**
- ***Product Owner*** **(P**
- **Time**

Artefatos no SCRUM

- ***Product Backlog***
- ***Sprint Backlog***
- ***Burndown* da *Sprint***
- ***Impedimentos***

Práticas do SCRUM

- ***Release Planning Meeting***
- ***Sprint***
- ***Sprint Planning Meeting***
- ***Daily Scrum Meeting***
- ***Sprint Review***
- ***Sprint Retrospective***

Ciclo de vida do SCRUM

- **Planejamento (*Pre-game phase*): Nesta fase é produzido o *Product Backlog,* definido o cronograma inicial e estimado o custo**
- **Desenvolvimento (*game phase*): Nesta fase o sistema é desenvolvido em Sprints**
- **Releasing (*post-game phase)*: Nesta fase é realizada a preparação para a entrega do software ao cliente**

Feature Driven Development (FDD)

- Resultados úteis a cada duas semanas ou menos
- Blocos bem pequenos de funcionalidade valorizados pelo cliente, chamados "*features*“
- Planejamento detalhado e guia para medição
- Monitoramento detalhado dentro do projeto, com resumos de alto nível para clientes e gerentes
- Fornece uma forma de saber, dentro dos primeiros 10% de um projeto, se o plano e a estimativa são sólidos

Papéis no FDD

- **Gestor do Projeto: trata das questões financeiras e administrativas do projeto**
- **Chefe de *Design*: responsável por toda a arquitetura do projeto, bem como pelas sessões de design, nas quais apresenta seu entendimento ao time**
- **Gestor de Desenvolvimento: acompanha as atividades de desenvolvimento do código diariamente**
- **Programador Chefe: é responsável por uma equipe pequena no que se refere à divisão e atribuição de trabalho entre seus membros**

Papéis no FDD

- **Dono de Classe: responsável pela arquitetura, implementação, teste e documentação de uma determinada classe**
- **Especialista da Área: membro conhecedor do assunto sobre o qual a aplicação atuará**

Práticas no FDD

- **Modelagem de objeto do domínio: é construída, inicialmente, uma modelagem genérica com as funcionalidades do sistema**
- **Desenvolvimento por funcionalidade: as atividades a serem desenvolvidas devem ser analisadas com a perspectiva de verificar a possibilidade de serem recompostas em atividades menores**
- **Posse individual do código: uma funcionalidade ou um conjunto delas é delegado a determinado desenvolvedor e este se torna automaticamente responsável por tudo que estiver relacionado ao código**

Práticas no FDD

- **Equipes de funcionalidades: as equipes são montadas para atender determinada funcionalidade ou um pequeno conjunto delas**
- **Inspeções: possibilita um melhoramento do código no que se refere à redução de erros, melhor modelagem e fatores como legibilidade e alta coesão**
- **Gerência de configuração: atividade que busca manter o controle sobre o código fonte**
- ***Build* constante: deve existir sempre uma versão do sistema rodando numa máquina**
- **Visibilidade do progresso**

Ciclo de vida no FDD

- **Duas fases: Concepção/Planejamento e Construção**
- **Cinco processos: Desenvolver um modelo abrangente, Gerar uma lista de funcionalidades, Planejar por funcionalidade, Detalhar por funcionalidade e Construir por funcionalidade**