Aula 2

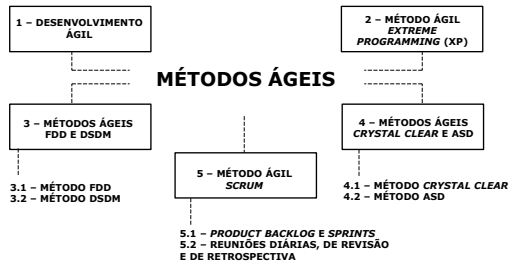
Engenharia de Software

Prof. Alex Mateus Porn

1

Conversa Inicial

2



3

Desenvolvimento ágil

4

- **"... quando modelos prescritivos são aplicados a sistemas de negócios de pequeno ou médio porte, a sobrecarga é tão grande que domina o processo de desenvolvimento de *software*" (Sommerville, 2018, p. 60)**
- **Mais tempo é investido nas decisões do que no desenvolvimento ou nos testes**
- **Se os requisitos mudam, retrabalho é necessário, e a especificação e o projeto têm de mudar**

5

- **A partir de 1990, começam a surgir os métodos ágeis para desenvolvimento de *software***
- **Em 2001, é criado o Manifesto para o Desenvolvimento Ágil de *Software* (*Agile Alliance*)**
- **Indivíduos e interações, mais que processos e ferramentas**
- ***Software* em funcionamento, mais que uma documentação abrangente**
- **Colaboração com o cliente, mais que negociação de contratos**
- **Responder a mudanças, mais que seguir um plano**

6

- **Incentiva a estruturação e as atitudes em equipe que tornam a comunicação mais fácil entre os membros**
- **Enfatiza a entrega rápida do *software* operacional e diminui a importância dos artefatos intermediários**
- **Assume o cliente como parte da equipe de desenvolvimento**
- **Reconhece que o planejamento em um mundo incerto tem seus limites e que o plano de projeto deve ser flexível (Pressman, 2011)**

7

- **"A agilidade pode ser aplicada a qualquer projeto de *software*" (Pressman, 2011, p. 83)**
  - **Seja projetada para que a equipe possa adaptar e alinhar tarefas**
  - **Possa conduzir o planejamento, compreendendo a fluidez de uma abordagem ágil**
  - **Possa eliminar tudo, exceto os artefatos essenciais, conservando-os enxutos**
  - **Enfatize a estratégia de entrega incremental**

8

- **O "Manifesto Ágil" estabelece doze princípios de agilidade para quem quiser ser ágil:**

1. **A prioridade é satisfazer o cliente por meio de entrega adiantada e contínua de *software***
2. **Acolha bem os pedidos de alterações, mesmo atrasados no desenvolvimento**
3. **Entregue *software* funcionando com frequência, de algumas semanas para alguns meses, com preferência a intervalos curtos**

9

4. **O pessoal comercial e os desenvolvedores devem trabalhar em conjunto**
5. **Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o apoio necessários**
6. **O método mais eficiente e efetivo de transmitir informações é uma conversa aberta, de forma presencial**
7. ***Software* em funcionamento é a principal medida de progresso**

10

8. **Os processos ágeis promovem desenvolvimento sustentável. Os envolvidos devem manter um ritmo constante**
9. **Atenção contínua à excelência técnica e aos bons projetos aumenta a agilidade**
10. **Simplicidade é essencial**
11. **Os melhores projetos emergem de equipes que se auto-organizam**
12. **A intervalos regulares, a equipe se avalia para ver como se tornar mais eficiente**

11

Método ágil *Extreme Programming* (XP)

12

Programação extrema – XP

- Surgiu nos EUA no final da década de 1990
- Preconiza mudanças incrementais e *feedback* rápido
- Considera a mudança como parte do processo
- Considera que pequenos ganhos a curto prazo pelo sacrifício da qualidade não são compensados pelas perdas a médio e longo prazo

13

Aplicação do método XP

- "Para aplicar o método XP, é necessário seguir uma série de práticas que dizem respeito ao relacionamento com o cliente, a gerência do projeto, a programação e os testes" (Wazlawick, 2013, p. 62)

14

Práticas para a aplicação do método XP

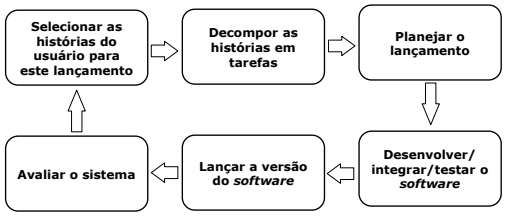
- Práticas XP
  - Jogo de planejamento
  - Metáforas
  - Equipes coesas
  - Reuniões em pé
  - *Design* simples
  - Versões pequenas
  - Ritmo sustentável

15

- Posse coletiva
- Programação em pares
- Padrões de codificação
- Testes de aceitação
- Desenvolvimento orientado a teste
- Refatoração
- Integração contínua

16

Ciclo de lançamento de *releases* do método XP



Fonte: elaborado com base em Sommerville, 2018, p. 62.

17

Métodos ágeis *Feature Driven-Development* (FDD) e *Dynamic Systems Development Method* (DSDM)

18

Desenvolvimento dirigido por funcionalidade – FDD

- Criado em 1997
- Enfatiza o uso de orientação a objetos
- Foca no desenvolvimento por funcionalidades
- Planejamento incremental e iterativo
- Integração contínua das funcionalidades
- Teste de *software*

19

- Dividido em duas fases
  - Concepção e planejamento
    - Prazo de execução entre uma e duas semanas
  - Construção
    - Ciclos iterativos de uma a duas semanas

20

Fase de concepção e planejamento do método FDD

- Concepção e planejamento
  - Desenvolver modelo abrangente – DMA
  - Construir lista de funcionalidades – CLF
  - Planejar por funcionalidade – PPF

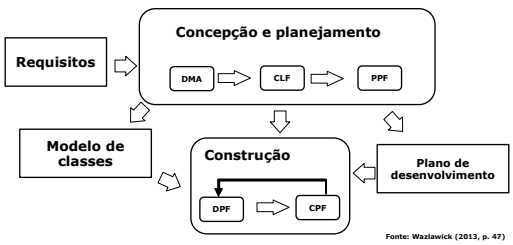
21

Fase de construção do método FDD

- Construção
  - Detalhar por funcionalidade – DPF
  - Construir por funcionalidade – CPF

22

Ciclo de execução do método FDD



Fonte: Wazlawick (2013, p. 47)

23

Desenvolvimento de sistemas dinâmicos – DSDM

- Planejamento incremental e iterativo
- Integração contínua das funcionalidades
- Teste de *software*
- Participação ativa do usuário

24

Desenvolvimento de sistemas dinâmicos – DSDM

- Dividido em três fases:
  - Pré-projeto
  - Ciclo de vida
  - Pós-projeto

25

- "Fundamenta-se no princípio de Pareto ou 80/20. Procura iniciar pelo estudo e pela implementação dos 20% dos requisitos mais determinantes para o sistema. Essa prática é compatível com abordar inicialmente requisitos mais complexos ou de mais alto risco" (Wazlawick, 2013, p. 52)

26

Princípios do método DSDM

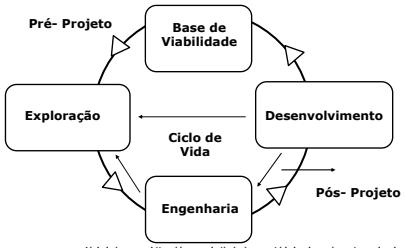
- Envolvimento do usuário
- Autonomia dos desenvolvedores
- Entregas frequentes
- Eficácia das entregas
- *Feedback* dos usuários

27

- Reversibilidade de ações
- Ritmo sustentável
- Previsibilidade das ações
- Ausência de testes no escopo
- Comunicação entre os envolvidos

28

Ciclo de desenvolvimento DSDM



Link da imagem: https://www.whatissixsigma.net/dsdm-dynamic-systems-development-method/

29

Métodos ágeis *Crystal Clear* e *Adaptive Software Development* – ASD

30

Método *crystal clear*

- Criado em 1997
- Adequado para equipes pequenas (máximo oito)
- Equipes compostas por um *designer* líder e entre dois e sete programadores
- Uso de "radiadores" de informação
- Fácil acesso a especialistas de domínio
- Eliminação de distrações
- Cronograma de desenvolvimento e ajuste do método quando necessário

31

Ciclo de vida do método *crystal clear*

- Organizado em três níveis:
  1. Iteração
     - Composto por estimação, desenvolvimento e celebração. Costuma durar poucas semanas
  2. Entrega
     - Formado por várias iterações. Ciclos máximos de dois meses
  3. Projeto
     - Formado pelo conjunto de todas as entregas

32

Sete pilares do método *crystal clear*

- Pilares *crystal clear*
  - Entregas frequentes
  - Melhoria reflexiva
  - Comunicação osmótica
  - Segurança pessoal
  - Foco
  - Acesso fácil a especialistas
  - Ambiente tecnologicamente rico

33

Desenvolvimento de software adaptativo – ASD

- Tem como base sistemas adaptativos complexos
- Enxerga o processo de desenvolvimento de *software* com
  - Agentes
    - Clientes, desenvolvedores e usuários
  - Ambientes
    - Organizacional, tecnológico e de processos

34

Fases do método ASD

- Fundamenta-se em desenvolvimento cíclico iterativo baseado em três grandes fases
  - Especular
  - Colaborar
  - Aprender

35

Fase de especulação ASD

- O projeto é iniciado
- O planejamento é conduzido em ciclos adaptáveis, com foco em:
  - Determinar o tempo de duração do projeto
  - Quantidade de ciclos e duração de cada um
  - Objetivos de cada ciclo
  - Componentes a serem desenvolvidos, tecnologias necessárias e listas de tarefas

36

Fase de colaboração ASD

- A equipe tem como foco:
  - Realizar as atividades que podem ser mais previsíveis
  - Realizar as atividades que são naturalmente mais incertas
- A partir dessa colaboração, vários componentes serão desenvolvidos de forma concorrente

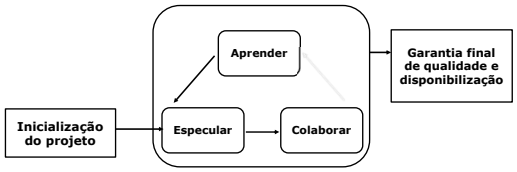
37

Fase de aprendizado ASD

- Revisão de qualidade
- Exige repetidas revisões de qualidade
- Repetidos testes de aceitação
- Necessária a presença do cliente e de especialistas do domínio

38

Ciclo de execução do método ASD



Fonte: Wazlawick, 2013, p. 74.

39

Método ágil *Scrum*

40

Scrum

- Tem como foco a gestão de projetos de *software*
- Tem como um dos conceitos mais importantes o *sprint*
- Pode ser integrado a outros métodos ágeis com facilidade
- Por isso, pode-se dizer que é um dos métodos ágeis mais utilizados

41

Resumidamente

- O *scrum* é aplicado em três fases
  - Primeira fase
    - Planejamento geral. Estabelecem-se os objetivos gerais do projeto e da arquitetura
  - Segunda fase
    - Execução dos ciclos de *sprint*. Cada ciclo desenvolve um incremento do sistema

42

- **Terceira fase**
  - **Encerra-se o projeto. Completa-se a documentação. Avaliam-se as lições aprendidas com o projeto**

43

Papéis do *scrum*

- ***Scrum master***
  - **Facilitador e solucionador de conflitos. Responsável por manter o time *scrum* em um ambiente propício para concluir o projeto**
- ***Product owner***
  - **Representa a voz do cliente. Responsável pelo projeto em si. Indica quais são os requisitos mais importantes a serem tratados em cada *sprint***

44

- ***Scrum team***
  - **Trata-se da equipe de desenvolvimento. Todos interagem para desenvolver o produto em conjunto**

45

Product backlog

- **Refere-se às funcionalidades a serem implementadas em cada projeto**

Product backlog					
Cód.	Nome	Importância	Estimativa de esforço	Como demonstrar	Notas
1	Depósito	30	5	Logar, abrir página de depósito, depositar R$10,00, ir para a página de saldo e verificar que ele aumentou R$10,00	Precisa de um diagrama de sequência UML. Não há necessidade de se preocupar com criptografia por enquanto
2	Ver extrato	10	8	Logar, clicar em "Transações". Fazer um depósito. Voltar para "Transações", ver que o depósito apareceu	Usar paginação para evitar consultas grandes ao BD. Design similar para visualizar a página de usuário

46

Sprints

- **Ciclo de desenvolvimento que, em geral, vai de duas semanas a um mês**
- **Os elementos do *product backlog* a serem implementados são priorizados e transferidos para o *sprint backlog***
- **Mantém-se o *sprint backlog* atualizado, indicando as tarefas já concluídas e aquelas ainda por concluir**

47

Reuniões

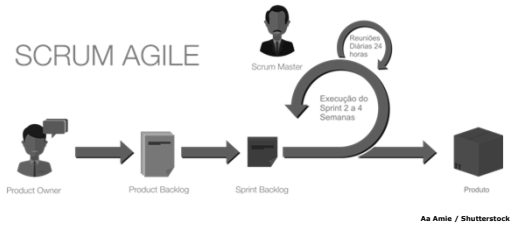
- **Diárias**
  - **Realizadas a cada 24 horas, em pé**
  - **Atualizar o time sobre o andamento do *sprint***
- **Revisão**
  - **Realizadas ao final de cada *sprint***
  - **Avalia o produto do trabalho**

48

- **Retrospectiva**
  - **Tem como objetivo avaliar os *sprints* concluídos**
  - **Avalia o que deu certo e errado, o que pode ser evitado e feito para reduzir erros, e identifica melhorias**

49

Ciclo *scrum*



Aa Amie / Shutterstock

50

Referências

51

- **PRESSMAN, R. S. Engenharia de *software*: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.**
- **SOMMERVILLE, I. Engenharia de *software*. 10. ed. São Paulo: Pearson Education do Brasil, 2018.**
- **WAZLAWICK, R. S. Engenharia de *software*: conceitos e práticas. São Paulo: Elsevier, 2013.**

52