Aula 4

Qualidade de Software

Profª Maristela Weinfurter

Conversa Inicial

Agilidade e qualidade

Manifesto Ágil | Quality Agile | Métricas Ágeis

- Manifesto Ágil
  - Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado
  - Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente

- Manifesto Ágil
  - Entregar frequentemente um software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo
  - Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto

- Manifesto Ágil
  - Construir projetos em torno de indivíduos motivados. Dar a eles o ambiente e o suporte necessário e confiar neles para fazer o trabalho
  - O método mais eficiente e eficaz de transmitir informações para uma equipe de desenvolvimento e entre ela é através de conversa face a face

- **Manifesto Ágil**
  - **Software funcionando é a medida primária de progresso**
  - **Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente**

- **Manifesto Ágil**
  - **Contínua atenção à excelência técnica e ao bom design aumenta a agilidade**
  - **Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial**

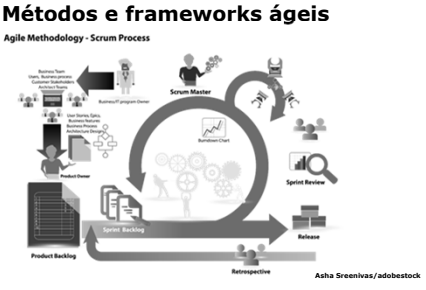
- **Manifesto Ágil**
  - **As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis**
  - **Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e, então, refina e ajusta seu comportamento de acordo com isso**

Métodos e frameworks ágeis

Métodos e frameworks ágeis

- **SCRUM**
- **KANBAN**
- **Extreme Programming (XP)**
- **FDD (Feature Driven Development)**
- **DSDM (Dynamic System Development Model)**
- **BDD (Behavior Driven Development)**
- **TDD (Test Driven Development)**

Métodos e frameworks ágeis



Asha Sreenivas/adobestock

Métodos e frameworks ágeis

ra2 studio/Shutterstock

- **Scrum**
  - **Iteração/sprint**
  - **Equipe scrum**
    - **Devs**
    - **Eng. qualidade**
    - **Documentação técnica**
    - **PO (proprietário do produto)**
    - **SM (scrum master)**

- **SCRUM**
  - **DoR (definição de pronto)**
  - **DoD (checklist da definição de pronto)**
  - **Critérios de aceitação (marcação do que foi aceito depois de pronto)**

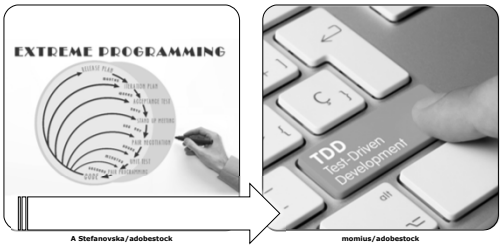
- **Scrum**
  - **Refinamento do backlog**
  - **Planejamento da sprint (iteração)**
  - **Levantamento diário (daily)**
  - **Demonstração de fim de sprint**
  - **Retrospectiva da sprint**
  - **Priorização do backlog**

- **Requisitos das lideranças para que o modelo Ágil seja bem-sucedido**
  - **Comunicar-se claramente com todos os níveis de sua organização para garantir que a lógica da mudança seja compreendida**
  - **Presença garantida em todas as reuniões do modelo adotado (daily, retro, planning, etc.)**

- **Requisitos das lideranças para que o modelo ágil seja bem-sucedido**
  - **Compromisso pelo exemplo**
  - **Reforço constante sobre a mudança**
  - **Abertura para feedback construtivo**
  - **Acreditar no modelo adotado**

TDD - Test Drive Development

TDD - Test Drive Development



A Stefanovska/adobestock momius/adobestock

TDD – Test Drive Development

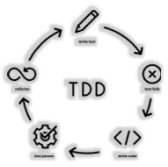
- Com TDD, conseguimos
  - Código limpo (sem código desnecessário e/ou duplicado)
  - Código-fonte dos testes como documentação dos casos de testes
  - Código confiável, logo, com melhor qualidade

TDD – Test Drive Development

- Com TDD, conseguimos
  - Suporte para teste de regressão
  - Ganho de tempo na depuração e correção de erros
  - Desenvolvimento refatorado constantemente
  - Baixo acoplamento do código

TDD - Test Drive Development

- O ciclo de desenvolvimento é composto por sinalizadores "red", "green" e "refactor"
  - Escrita do teste inicial. Flag red
  - Adição de nova funcionalidade
  - Execução do teste passar. Flag green
  - Refatoração do código
  - Escrita do próximo teste



Vector street/Shutterstock

DDD – Domain-Driven Design

DDD – Domain-Driven Design

- O Domain-Driven Design (DDD) é uma filosofia de desenvolvimento projetada para gerenciar a criação e manutenção de software escrito para domínios de problemas complexos

Dmitry Kovalchuk/Shutterstock

DDD – Domain-Driven Design

- Agrega uma coleção de padrões, princípios e práticas, que podem ser aplicados ao projeto de software para gerenciar a complexidade

Dmitry Kovalchuk/Shutterstock

DDD – Domain-Driven Design

- O DDD é arquitetonicamente agnóstico, pois não há um único estilo de arquitetura que você deva seguir para implementá-lo. Os estilos de arquitetura podem variar porque devem ser aplicados no nível de contexto limitado e não no nível do aplicativo

Dmitry Kovalchuk/Shutterstock

DDD – Domain-Driven Design

- Uma linguagem ubíqua é usada para vincular o modelo de análise ao modelo de código para que a equipe de desenvolvimento e os especialistas de domínio colaborem no design de um modelo

Dmitry Kovalchuk/Shutterstock

- O DDD não dita nenhum estilo de arquitetura específico para desenvolvimento, apenas garante que o modelo seja mantido isolado das complexidades técnicas para que possa se concentrar nas preocupações da lógica do domínio

Dmitry Kovalchuk/Shutterstock

BDD – Behavior Driven Development

BDD – Behavior Driven Development

- BDD é um processo de desenvolvimento de software, baseado no Test-Driven Development (TDD), que se concentra em capturar o comportamento de um sistema e, em seguida, direcionar o design de fora para dentro

Zhanna Hapanovich/Shutterstock

BDD – Behavior Driven Development

- BDD não se concentra nos aspectos técnicos de um aplicativo
- BDD se concentra no comportamento do software

Zhanna Hapanovich/Shutterstock

BDD – Behavior Driven Development

- Esse modelo utiliza sua própria forma de UL para especificar requisitos (uma linguagem de análise)
- Conhecida como GWT (given, when, then)

Zhanna Hapanovich/Shutterstock

BDD – Behavior Driven Development

- O uso desse método de captura de requisitos remove a ambiguidade que a documentação tradicional de requisitos pode resultar, ao mesmo tempo em que enfatiza fortemente a linguagem do domínio

Zhanna Hapanovich/Shutterstock

BDD – Behavior Driven Development

- Os recursos e cenários são um produto da colaboração entre a equipe de desenvolvimento e especialistas de negócios e podem ajudar a moldar o UL

Zhanna Hapanovich/Shutterstock

BDD – Behavior Driven Development

- BDD auxilia a condução dos requisitos e comportamentos que nosso software deve ter

Zhanna Hapanovich/Shutterstock

UX e o desenvolvimento ágil

UX e o desenvolvimento ágil

- Equipes multidisciplinares, idealmente colocalizadas
- Uma forte ênfase na comunicação interpessoal
- A substituição de requisitos formalmente documentados por um backlog continuamente gerenciado e priorizado, com as histórias de usuários e foco na prototipagem

UX e o desenvolvimento ágil

- Desenvolvimento iterativo organizado em períodos fixos curtos, conhecidos como iterações, sprints ou timeboxes
- Entrega incremental frequente de software o mais ágil possível

Prototipação

mangpor2004/adobestock

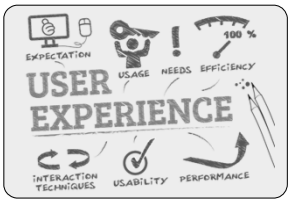
Prototipação

123levit/adobestock

Prototipação

- Baixa fidelidade (wireframe)
- Média fidelidade (mockup)
- Alta fidelidade (HTML, CSS, Bootstrap, JSX etc.)

Usabilidade e testes de usabilidade



Trueffelpix/adobestock

Coloures-Pic/adobestock

Usabilidade e testes de usabilidade

- Avaliação heurística
- Inspeção por checklists
- Testes de percurso
- Entrevistas e questionários
- Percurso cognitivo

- As dez heurísticas de Nielsen
  - Visibilidade do status do software
  - Correspondência entre o software e a vida real
  - Liberdade e controle do usuário
  - Consistência e padrões
  - Prevenção de erros

- As dez heurísticas de Nielsen
  - Reconhecimento e não lembrança
  - Flexibilidade e eficiência
  - Estética e design minimalista
  - Auxílio a usuários no reconhecimento, no diagnóstico e na recuperação de erros
  - Ajuda e documentação