

## Aula 6

### Metodologias Ágeis

Prof. Manoel Flavio Leal

1

### Conversa Inicial

2

### Estudaremos nesta aula

- Gestão ágil de projetos
- Planejamento e definição de requisitos
- Gerenciamento da equipe e progresso
- Configuração de projetos ágeis



3

### Objetivos

- Aprender a gerenciar projetos ágeis de forma eficiente
- Promover a colaboração e comunicação na equipe
- Dominar ferramentas e práticas de integração contínua

4

### Planejamento do Projeto

5

### Planejamento ágil

- Combina abordagens ágeis e gerenciamento de projetos tradicionais
- Permite iteração, ajuste do plano e adaptação a mudanças



6

### Componentes do planejamento ágil

- Definição de um objetivo do cliente
- Evitar detalhes desnecessários
- Realização de entregas frequentes e iterativas
- Estabelecimento de intervalos de datas em vez de estimativas fixas



Joana Silva / Shutterstock

7

- Foco no trabalho em si, não apenas no executor
- Ausência de uma fase separada para garantia de qualidade
- Utilização de planos de duas camadas
- Embasamento em dados
- Ênfase na comunicação, flexibilidade, praticidade e satisfação do cliente

8

### Agile planning onion

- Descreve os diferentes níveis de planejamento em metodologias ágeis
- Camadas internas representando detalhamento e horizontes de tempo curtos, e camadas externas representando níveis mais amplos e de longo prazo
- Amplamente adotado na comunidade ágil e considerado eficaz para abordar o planejamento em abordagens ágeis



Henr\_Abrine / Shutterstock

9

### Camadas do agile planning onion

- Estratégia (*strategy*): decisões estratégicas, objetivos de negócio e metas estratégicas
- Portfólio (*portfolio*): planejamento em um nível mais amplo, seleção e priorização de projetos
- Produto (*product*): definição e gerenciamento do produto, identificação de recursos e funcionalidades



danah / Shutterstock

10

- Release: planejamento dos lançamentos do produto, definição de prazos e gestão de dependências
- Iteração (*iteration*): planejamento das iterações, seleção de histórias de usuário e definição de tarefas
- Daily: definição das atividades diárias para concluir as tarefas da iteração

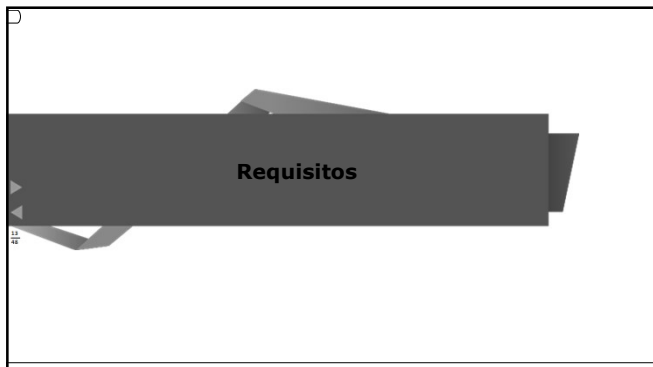


danah / Shutterstock

11

- O planejamento ágil combina métodos ágeis com gerenciamento tradicional, permitindo ajustes durante a execução
- O *agile planning onion* representa visualmente os níveis de planejamento, de estratégico a operacional

12



13

### Requisitos e a modelagem ágil

- A abordagem ágil valoriza a flexibilidade e a entrega contínua de valor em detrimento da documentação estática
- Compreender requisitos é um processo constante e colaborativo que envolve todas as partes interessadas
- Colaboração, comunicação e técnicas como histórias de usuário garantem alinhamento efetivo e adaptação dos requisitos ao longo do projeto



14

### Exemplos de requisitos

- História da usuário
  - "Como um cliente, eu quero poder adicionar itens ao meu carrinho de compras para facilitar o processo de compra on-line"
  - Essa história captura um requisito específico do sistema, focando nas necessidades do cliente e fornecendo uma base clara para o desenvolvimento

15

- Critérios de aceitação
  - Capacidade de adicionar itens ao carrinho, a exibição correta dos itens selecionados e a atualização em tempo real do total do carrinho
  - Definem os critérios pelos quais uma funcionalidade será considerada concluída e aceita pelo cliente

16

- Requisitos não funcionais
  - O sistema deve suportar um número mínimo de usuários simultâneos sem comprometer o desempenho
  - Esses requisitos abrangem aspectos como desempenho, segurança, usabilidade, escalabilidade

17

- Priorização de requisitos
  - História 1: "como usuário, desejo poder realizar *login* na plataforma para acessar meus dados pessoais"
  - História 2: "como usuário, desejo poder pesquisar produtos por categoria para encontrar o que estou procurando"
  - Com base na importância e no valor percebido pelo cliente, a equipe decide priorizar a história 1, pois o acesso seguro à plataforma é considerado fundamental para os usuários, antes de implementar a história 2

18


### Principais técnicas

- Entrevistas
- Workshops colaborativos
- Prototipagem
- Análise de documentos e artefatos existentes

delwar hassain / shutterstock linear\_design / shutterstock alx1618 / shutterstock alexander lysenko / shutterstock

19

- Observação de usuário
- Storytelling
- Análise e feedback e métricas




treetly / Shutterstock  
Zatris.Creative / Shutterstock  
Golden Silkorka / Shutterstock

20

### Benefícios


- Compreensão clara das necessidades dos *stakeholders*
- Redução de ambiguidades e inconsistências
- Validação contínua dos requisitos



eamesBot / Shutterstock

21

- Melhor colaboração e alinhamento entre as partes interessadas
- Agilidade e flexibilidade na adaptação aos requisitos em evolução
- Entrega de valor aos clientes e usuários finais



GOLDMANN / Shutterstock

22

### Conversão de Requisitos em Classe

23

### Convertendo requisitos em classes

- Etapa crucial no desenvolvimento de *software* orientado a objetos
- Requisitos analisados e transformados em classes
- Classes são unidades fundamentais do sistema
- A conversão permite representação clara e organizada para o design e implementação do *software*

24

## Passos para conversão

- Identificação e análise de requisitos
- Criação de modelos conceituais
- Mapeamento de requisito para classe



VectorMine / Shutterstock

25

- Definição de atributos e métodos
- Validação e verificação
- Gerenciamento de mudanças



Drawing Station / Shutterstock

26

## Exemplo de conversão

Histórias	Critérios de aceitação
História do usuário 1: Como usuário, desejo criar uma tarefa para poder acompanhar minhas atividades	Como usuário, quero poder fornecer um título, descrição e data de vencimento para a tarefa
História de usuário 2: Como usuário, quero atribuir uma prioridade às tarefas para poder identificar as mais importantes	Como usuário, quero poder escolher entre as opções de prioridade "baixa", "média" ou "alta" para cada tarefa
História de usuário 3: Como usuário, desejo marcar uma tarefa como concluída para poder acompanhar o progresso das minhas atividades	Como usuário, quero poder indicar que uma tarefa foi concluída
História de usuário 4: Como usuário, quero categorizar as tarefas em diferentes projetos para poder organizar melhor minhas atividades	Como usuário, quero poder associar uma tarefa a um projeto específico
História de usuário 5: Como usuário, desejo adicionar comentários às tarefas para poder registrar informações adicionais sobre elas	Como usuário, quero poder adicionar comentários a uma tarefa existente

27

## Exemplo: classes identificadas

Tarefa
- título : char
- descrição: char
- dataVencimento: char
- prioridade: char
- concluida: char
- comentarios: char
+ incluir() : void
+ excluir() : void
+ consultar() : void
+ alterar() : void

Projeto
- nome : char
- tarefas: Projeto
+ incluir() : void
+ excluir() : void
+ consultar() : void
+ alterar() : void

28

## Validação e verificação de requisitos convertidos em classes

- Garantir que as classes correspondam aos requisitos corretamente – validar
- Assegurar que as classes estejam em conformidade com os requisitos – verificar
- Revisões de código para identificar erros e inconsistências
- Realização de testes abrangendo diferentes cenários



Bakhtiar Zein / Shutterstock

29

- Mapear requisitos para classes é crucial no desenvolvimento orientado a objetos
- Garante representação clara e organizada dos requisitos no código
- Validação e verificação garantem que as classes atendam aos requisitos
- Resulta em *software* confiável e aderente aos requisitos

30

## Gerenciamento de Projetos Ágeis

## Gerenciamento ágil

- Abordagem adaptativa e colaborativa
- Ênfase na flexibilidade e capacidade de resposta a mudanças
- Mentalidade iterativa e incremental
- Principais metodologias: Scrum e Kanban
- Valorização da colaboração, *feedback* contínuo e melhoria contínua

## Equipe de projeto ágil

- Auto-organização e autogerenciamento
- Reuniões regulares para planejamento, revisão e ajustes
- Colaboração e comunicação eficazes são fundamentais

## Papéis na equipe ágil

- *Product owner*
- *Scrum master*
- Equipe de desenvolvimento



## Acompanhamento do que está sendo produzido

- Quadro Kanban para visualizar o fluxo de trabalho
- Burnup Chart para comparar o trabalho planejado com o concluído



- Reuniões diárias (*daily stand-ups*)
- Revisão de *sprint* para avaliar entregas
- Retrospectiva para refletir sobre o processo



### Uso de métricas ágeis

- Métricas para medir o progresso de forma objetiva
- **Velocity**: medida da quantidade de trabalho concluído em cada *sprint*
- **Lead time**: tempo necessário para completar um item de *backlog*
- Defeito ou taxa de *bug*: medida da quantidade de defeitos encontrados



37

### Ajustes e mudanças durante o projeto

- Flexibilidade para lidar com mudanças nos requisitos
- Adaptação do planejamento e prioridades ao longo do projeto
- Cerimônias e práticas específicas para gerenciar mudanças

38

### Mínimo produto viável (MVP)

- Objetivo: obter *feedback* e aprendizado rápido
- Lançamento do produto com o mínimo de recursos necessários



39

- Evita desperdício de tempo e recursos em funcionalidades não valorizadas
- Reduz riscos e fornece *insights* sobre o desempenho do produto

### MVP



40

- O gerenciamento de projetos ágeis é adaptativo e colaborativo
- Equipe auto-organizada e autogerenciada
- Acompanhamento contínuo do progresso com métricas ágeis
- Flexibilidade para ajustes e mudanças ao longo do projeto
- MVP permite validar e aprimorar o produto de forma iterativa

41

### Configuração de Projetos Ágeis

42

### Configurando o projeto

- Alinhamento estratégico com objetivos da organização
- Definição clara de papéis e responsabilidades
- Flexibilidade para lidar com mudanças e responder rapidamente
- Entrega contínua de valor ao cliente



ankastudio22 / Shutterstock

43

### Infraestrutura e ambientes de desenvolvimento

- Servidor de controle de versão (ex.: Git)
- Plataforma de hospedagem de repositórios (ex.: GitHub)
- Ferramenta de gerenciamento de projetos (ex.: Jira)
- Ferramentas de comunicação (ex.: Slack)

44

### Equipe multifuncional

- Desenvolvedores, designers de UI e UX
- Definição clara de papéis e responsabilidades
- *Product owner*, *scrum master* e equipe de desenvolvimento



Viktoria Kurpas / Shutterstock

45

### Ambientes de testes e integração contínua

- Ambientes separados para diferentes tipos de testes
- Ferramentas de automação de testes de UI
- Práticas de integração contínua para compilar e testar automaticamente o código

46

### Iterações e entregas frequentes

- *Sprints* de curta duração (ex.: duas semanas)
- Reuniões diárias (*daily stand-ups*) para alinhar atividades
- Revisões de *sprint* para coletar *feedback* dos *stakeholders*
- Ajuste de prioridades com base no *feedback* recebido

47

- Adaptar-se às mudanças e prioridades é essencial para o sucesso ágil
- A comunicação eficiente fortalece a colaboração na equipe
- Entrega contínua de valor mantém os clientes satisfeitos
- *Agile planning onion* e conversão de requisitos em classe impulsionam o planejamento eficaz
- Configuração adequada e MVP garantem resultados extraordinários

48



