



METODOLOGIAS ÁGEIS

AULA 6



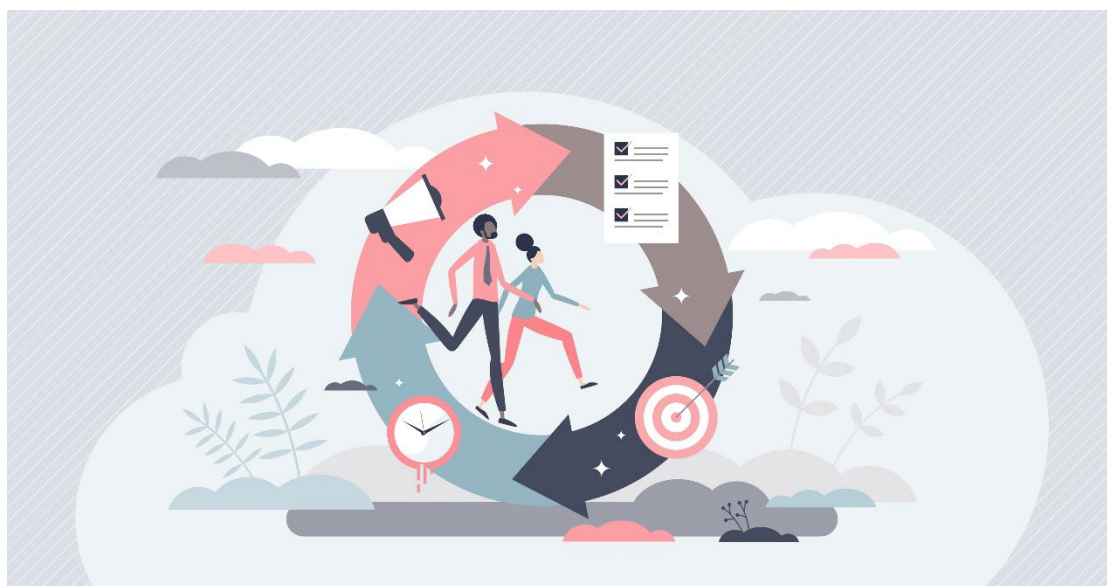
Prof.^a Mariana Lucia Kalluf Dakkache Leal

Gestão ágil (Gerenciamento e configuração)

Nesta abordagem, veremos a Gestão Ágil, focando no gerenciamento e configuração de projetos ágeis. Exploramos o planejamento do projeto, a definição de requisitos e sua conversão em código. Discutimos também o gerenciamento da equipe, acompanhamento do progresso e como lidar com mudanças, além do conceito de MVP (*Minimum Viable Product*). Por fim, tratamos da configuração de projetos ágeis, como a escolha de ferramentas e a integração contínua. O objetivo é aprender a gerenciar projetos ágeis de forma eficiente, promovendo a colaboração e entregando um produto de qualidade.

TEMA 1 – PLANEJAMENTO DO PROJETO

O planejamento ágil, que é utilizado para traçar o roteiro de um projeto com base nos métodos ágeis, é uma estratégia que combina abordagens ágeis com gerenciamento de projetos tradicionais. Auxilia organizações em transição da gestão tradicional para a ágil. Esse método permite a iteração e constante ajuste do plano, visando investir tempo no planejamento no momento mais adequado e adaptar-se facilmente a mudanças durante sua execução.



Créditos: VectorMine/ Shutterstock.



Os componentes do planejamento ágil incluem:

1. **Definição de um objetivo do cliente:** entender as necessidades, expectativas e requisitos do cliente para orientar o desenvolvimento do produto ou projeto.
2. **Evitar detalhes desnecessários:** ao contrário de abordagens tradicionais de planejamento, o planejamento ágil busca evitar o excesso de detalhes desnecessários. A ideia é focar nas informações essenciais para orientar o trabalho, evitando a sobrecarga de informações, que podem levar a atrasos e falta de flexibilidade.
3. **Realização de entregas frequentes e iterativas:** o trabalho é dividido em iterações ou sprints, em que pequenas entregas funcionais são realizadas de forma frequente e iterativa. Isso permite obter feedback rápido dos clientes e usuários, além de possibilitar ajustes e melhorias contínuas ao longo do tempo.
4. **Estabelecimento de intervalos de datas em vez de estimativas fixas:** em vez de definir estimativas de datas fixas para as entregas, o planejamento ágil utiliza intervalos de datas. Isso reconhece a natureza iterativa e adaptativa do trabalho ágil, permitindo ajustes com base nas necessidades e no progresso real alcançado.
5. **Foco no trabalho em si, não apenas no executor:** enfatiza o trabalho em si, ou seja, as tarefas e atividades necessárias para entregar o produto. Em vez de se concentrar apenas na atribuição de responsabilidades aos membros da equipe, o planejamento ágil valoriza a colaboração e a cooperação entre todos os envolvidos no projeto.
6. **Ausência de uma fase separada para garantia de qualidade:** diferentemente de abordagens tradicionais, que têm uma fase separada para a garantia de qualidade, no planejamento ágil, a qualidade é incorporada ao longo de todo o processo. Testes contínuos, revisões e melhorias são realizados durante as iterações, garantindo que a qualidade seja uma preocupação constante.
7. **Utilização de planos de duas camadas:** utiliza uma abordagem de planos de duas camadas. A primeira camada é um plano de alto nível, que abrange o escopo geral e as principais entregas do projeto. A segunda camada é um plano detalhado, que é revisado e atualizado regularmente durante as iterações.



8. **Embasamento em dados:** fundamentado em dados e informações reais. Isso significa que as decisões de planejamento são embasadas em feedback, métricas de desempenho e outras informações relevantes. A utilização de dados ajuda a orientar o planejamento de forma mais objetiva e informada.
9. **Ênfase na comunicação, flexibilidade, praticidade e satisfação do cliente:** o planejamento ágil coloca grande ênfase na comunicação efetiva entre as partes interessadas, permitindo um alinhamento contínuo e a compreensão das expectativas do cliente. Além disso, o planejamento ágil valoriza a flexibilidade para responder às mudanças, a praticidade em adotar abordagens simples e eficazes, e a satisfação do cliente como um dos principais objetivos do projeto.

1.1 *Agile planning onion*

O *Agile Planning Onion* é um conceito que foi desenvolvido para descrever os diferentes níveis de planejamento em metodologias ágeis. O *Agile Planning Onion* é uma representação visual que evoluiu ao longo do tempo para ilustrar os diferentes níveis de planejamento em abordagens ágeis.

A analogia com uma cebola, na qual as camadas internas representam níveis de detalhamento e horizontes de tempo mais curtos, enquanto as camadas externas representam níveis mais amplos e de longo prazo, tornou-se uma forma popular de visualizar o planejamento ágil.

Esse conceito foi amplamente adotado e discutido na comunidade ágil, com várias adaptações e interpretações ao longo do tempo. Embora não haja um único autor ou evento específico associado ao *Agile Planning Onion*, ele é considerado uma representação útil e eficaz para entender e abordar o planejamento em metodologias ágeis.

SIX LEVELS OF PLANNING



Créditos: dizain/ Shutterstock.

As camadas do *Agile Planning Onion* são as seguintes:

1. **Estratégia (*strategy*):** nessa camada, as decisões estratégicas são tomadas em relação ao projeto ou produto. É onde são definidos os objetivos de negócio e as metas estratégicas. Essa camada estabelece a direção geral e o propósito do projeto, fornecendo uma base para as decisões de planejamento subsequentes.
2. **Portfólio (*portfolio*):** nessa camada, o planejamento ocorre em um nível mais amplo, abrangendo o conjunto de projetos e iniciativas relacionados. É onde são selecionados e priorizados os projetos com base em critérios estratégicos. O planejamento no nível do Portfólio considera restrições de recursos, prioridades e alinhamento com a estratégia geral da organização.
3. **Produto (*product*):** o planejamento é voltado para a definição e gerenciamento do produto em si. O planejamento do Produto envolve a identificação dos principais recursos e funcionalidades, bem como a definição das necessidades dos usuários e clientes. Essa camada também aborda o mapeamento do *backlog* do produto, que é uma lista de itens a serem desenvolvidos.



4. **Release (release):** esta camada está focada no planejamento dos lançamentos do produto, as funcionalidades do produto são agrupadas em releases ou versões específicas, que são entregues aos usuários ou clientes em intervalos determinados. O planejamento do Release envolve a seleção de funcionalidades para cada lançamento, a definição de prazos e a gestão de dependências entre as funcionalidades.
5. **Iteração (iteration):** as funcionalidades são divididas em itens menores e mais detalhados, chamados de histórias de usuário. O planejamento da Iteração envolve a seleção das histórias de usuário a serem trabalhadas durante uma iteração específica, a definição das tarefas necessárias e a estimativa do esforço para cada tarefa.
6. **Daily (daily):** as equipes definem as atividades diárias que serão realizadas para concluir as tarefas atribuídas durante a iteração. O planejamento diário envolve a definição das tarefas específicas a serem realizadas por cada membro da equipe e a atualização constante do progresso durante o dia.

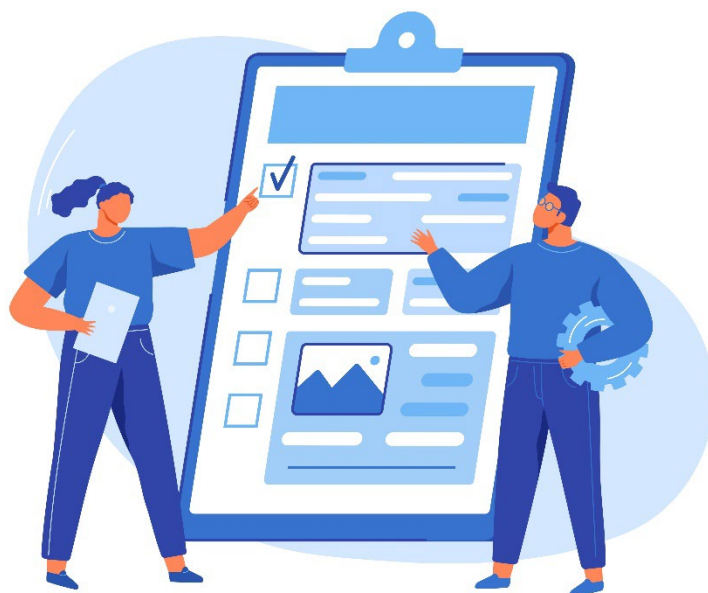
Essas camadas do *Agile Planning Onion* fornecem uma estrutura escalonável para o planejamento ágil, desde as decisões estratégicas até o nível operacional do dia a dia. Cada camada aborda um aspecto específico do planejamento, permitindo uma abordagem iterativa e adaptativa para entregar valor ao longo do tempo.

TEMA 2 – REQUISITOS (ALGUNS EXEMPLOS)

Em certas ocasiões, a documentação e os requisitos podem se tornar obsoletos antecipadamente à finalização do projeto (Jaqueira et al., 2012). Isso ocorre em virtude da natureza dinâmica dos projetos ágeis, nos quais as necessidades e os requisitos podem se alterar ao longo do tempo. A abordagem ágil valoriza a flexibilidade e a capacidade de resposta às mudanças, priorizando a entrega contínua de valor em detrimento de uma documentação ampla e estática. Desse modo, a compreensão dos requisitos em projetos ágeis é um componente crucial para o êxito de qualquer empreendimento ágil. Nesse contexto, compreender os requisitos transcende uma relação estática no início do projeto, sendo concebida como um processo constante e colaborativo. É imprescindível envolver todas as partes interessadas, desde os usuários finais

até os clientes e membros da equipe, a fim de compreender de forma clara e abrangente das necessidades e expectativas.

A colaboração e a comunicação desempenham um papel fundamental nesse procedimento, permitindo uma troca constante de informações e ideias entre todos os envolvidos. O enfoque principal consiste em identificar e priorizar os requisitos essenciais para proporcionar valor de maneira progressiva e iterativa. Por meio de técnicas como histórias de usuário, prototipagem e workshops colaborativos, é possível explorar e validar conjuntamente os requisitos, assegurando um alinhamento efetivo entre as partes interessadas. A flexibilidade surge como uma característica-chave, uma vez que os requisitos podem se desenvolver e se adaptar ao longo do projeto, conforme emergem novas percepções e mudanças.



Créditos: buart/ Shutterstock.

2.1 Exemplos de requisitos em projetos ágeis

Os requisitos em projetos ágeis são frequentemente expressos de forma clara e concisa, priorizados com base no valor para o cliente e estão sujeitos a mudanças contínuas à medida que o projeto evolui. A seguir, apresentaremos alguns exemplos de requisitos utilizados em projetos ágeis:

1. **História de usuário:** uma história de usuário é um exemplo comum de requisito em projetos ágeis. Ela descreve uma funcionalidade desejada do produto do ponto de vista do usuário final.



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

- 2. Critérios de aceitação:** os critérios de aceitação são requisitos adicionais que complementam uma história de usuário. Eles definem os critérios pelos quais uma funcionalidade será considerada concluída e aceita pelo cliente.

Por exemplo, para a história de usuário mencionada, os critérios de aceitação podem incluir a capacidade de adicionar itens ao carrinho, a exibição correta dos itens selecionados e a atualização em tempo real do total do carrinho.

- 3. Requisitos não funcionais:** além dos requisitos funcionais, os projetos ágeis também levam em consideração os requisitos não funcionais, que descrevem as características e restrições do sistema. Esses requisitos abrangem aspectos como desempenho, segurança, usabilidade, escalabilidade, entre outros.


Por exemplo, um requisito não funcional pode ser a exigência de que o sistema suporte um número mínimo de usuários simultâneos sem comprometer o desempenho.

- 4. Priorização de requisitos:** a priorização de requisitos é um exemplo importante em projetos ágeis. É a prática de ordenar os requisitos com base em sua importância e valor para o cliente. A equipe e o cliente colaboram para estabelecer uma ordem de prioridade, permitindo que as funcionalidades mais críticas sejam desenvolvidas primeiro. Isso garante que o trabalho seja direcionado para as áreas de maior impacto e valor, otimizando o retorno sobre o investimento do projeto.

Por exemplo: 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.

Priorização: com base na importância e valor percebido pelo cliente, a equipe decide priorizar a História 1, pois o acesso seguro à plataforma é

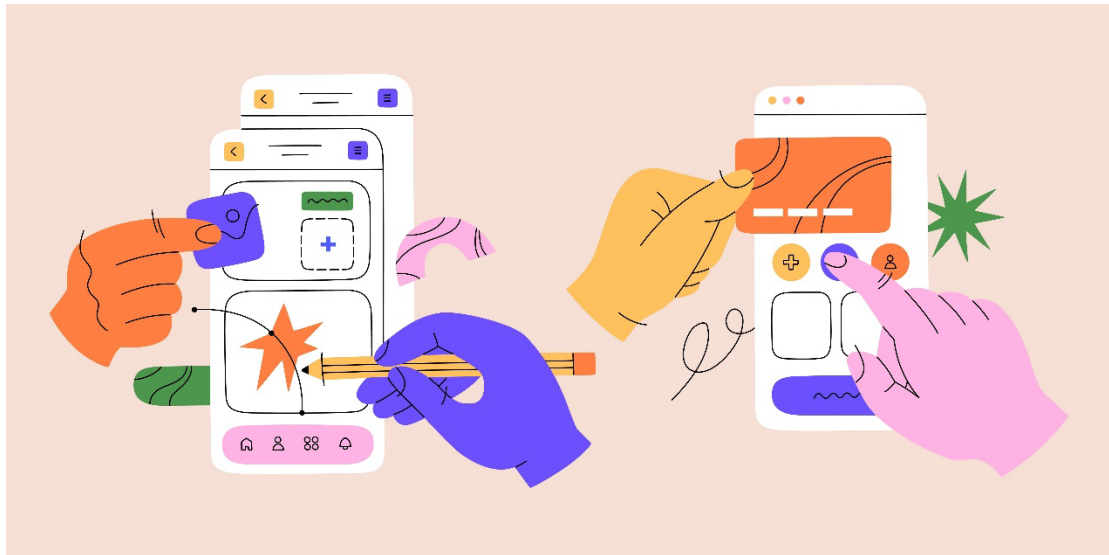


considerado fundamental para os usuários, antes de implementar a História 2.

2.2 Técnicas para levantamento de requisitos ágeis

O uso de técnicas para o levantamento de requisitos ágeis é fundamental por vários motivos:

- 1. Compreensão clara das necessidades dos stakeholders:** ajuda a obter uma compreensão mais precisa e completa do que é necessário para atender às suas demandas.
- 2. Redução de ambiguidades e inconsistências:** ao utilizar abordagens estruturadas, é possível obter uma definição mais clara e precisa dos requisitos, evitando interpretações divergentes e mal-entendidos.
- 3. Validação contínua dos requisitos:** evita atrasos e retrabalhos, garantindo que o produto atenda às necessidades e expectativas dos usuários.
- 4. Melhor colaboração e alinhamento entre as partes interessadas:** ao envolver todos os envolvidos, é possível obter uma visão mais abrangente e alinhada sobre os requisitos, evitando lacunas e divergências.
- 5. Agilidade e flexibilidade na adaptação aos requisitos em evolução:** por meio de abordagens iterativas e incrementais, as técnicas permitem ajustar e adaptar os requisitos à medida que novas informações são obtidas ou as prioridades mudam.
- 6. Entrega de valor aos clientes e usuários finais:** resulta em uma maior satisfação do cliente e na entrega de valor agregado desde as fases iniciais do projeto.



Créditos: ybealice/ Shutterstock.

Existem várias técnicas amplamente utilizadas para fazer o levantamento de requisitos em projetos ágeis. Algumas das principais técnicas incluem:

- 1. Entrevistas:** permitem obter *insights* diretos e esclarecer dúvidas, ajudando a capturar requisitos de forma mais precisa.
- 2. Workshops colaborativos:** envolvem a participação ativa de diferentes partes interessadas, como membros da equipe de desenvolvimento, clientes, usuários finais e especialistas em negócios. Durante os workshops, são realizadas atividades práticas, discussões em grupo e mapeamentos de processos para identificar e validar requisitos.
- 3. Prototipagem:** envolve a criação de representações visuais ou funcionais de partes do sistema. Os protótipos auxiliam na superação de diversos obstáculos encontrados no campo da engenharia de requisitos ágil (Käpyaho; Kauppinen, 2015), permitem uma visualização mais concreta das funcionalidades e interfaces do sistema, facilitando a compreensão e a validação dos requisitos pelos stakeholders.
- 4. Análise de documentos e artefatos existentes:** a análise de documentos, como documentos de visão, casos de uso, especificações existentes e outros artefatos relacionados ao projeto, pode fornecer informações valiosas sobre os requisitos e as necessidades do sistema.
- 5. Observação do usuário:** observar diretamente os usuários enquanto eles executam suas tarefas ou interagem com sistemas semelhantes pode revelar *insights* sobre seus comportamentos, necessidades e requisitos.



6. Storytelling: envolve a criação de histórias e cenários fictícios para descrever situações em que o sistema será utilizado. Essas histórias ajudam a capturar os requisitos de forma mais concreta, relacionando-os com casos de uso específicos.

7. Análise de feedback e métricas: pode ajudar a identificar melhorias e novos requisitos para o sistema.

TEMA 3 – CONVERSÃO DE REQUISITOS EM CLASSE

A conversão de requisitos em classes é uma etapa crucial no desenvolvimento de software orientado a objetos. Nesse processo, os requisitos funcionais e não funcionais identificados são analisados e transformados em classes, que são as unidades fundamentais de estrutura e comportamento do sistema. Essa conversão permite uma representação mais clara e organizada dos requisitos, fornecendo uma base sólida para o design e implementação do software.

Na sequência, apresentamos os passos geralmente seguidos para essa conversão:

1. Identificação e análise de requisitos:

- Compreenda os requisitos do sistema por meio de técnicas como entrevistas, prototipagem e workshops colaborativos;
- Priorize os requisitos com base no valor para o cliente, considerando sua importância e impacto no produto.

2. Criação de modelos conceituais:

- Desenvolva modelos conceituais que representem os principais conceitos e entidades envolvidos no sistema;
- Identifique os relacionamentos e associações entre esses conceitos.

3. Mapeamento de requisitos para classes:

- Mapeie os requisitos identificados nas classes do sistema;
- Identifique as classes necessárias para atender aos requisitos, levando em consideração as entidades e os comportamentos necessários.



4. Definição de atributos e métodos:

- Para cada classe, defina os atributos necessários para representar os dados relevantes;
- Identifique os métodos que serão responsáveis pelas operações e comportamentos da classe.

5. Validação e verificação:

- Realize uma validação dos requisitos convertidos em classes, assegurando que eles atendam aos objetivos do projeto;
- Verifique se as classes e seus atributos e métodos estão alinhados com os requisitos originais.

6. Gerenciamento de mudanças:

- Reconheça que os requisitos podem mudar ao longo do tempo e esteja preparado para ajustar as classes em resposta a essas mudanças;
- Adote práticas ágeis de gerenciamento de mudanças, como revisões contínuas dos requisitos e adaptação das classes conforme necessário.



Créditos: Danang Setiawan/ Shutterstock.



3.1 Exemplo de conversão de requisitos em classes

Vamos considerar um exemplo prático de conversão de requisitos ágeis para classes em um sistema de gerenciamento de tarefas. Suponha que os seguintes requisitos tenham sido identificados:

História de Usuário 1: como usuário, desejo criar uma tarefa para poder acompanhar minhas atividades.

Critérios de aceitação:

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.

Critérios de aceitação:

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.

Critérios de aceitação:

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.

Critérios de aceitação:

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.

Critérios de aceitação:

Como usuário, quero poder adicionar comentários a uma tarefa existente.

Agora, vamos converter esses requisitos em classes:

Classe Task:

- Atributos:

- title: string
- description: string
- dueDate: date
- priority: string
- completed: boolean

- comments: array

Classe Project:

- Atributos:

- name: string
- tasks: array

As classes refletem os requisitos apresentados anteriormente, permitindo que a equipe de desenvolvimento implemente as funcionalidades necessárias. A classe Task representa uma tarefa específica e possui atributos como "title" (título), "description" (descrição), "dueDate" (data de vencimento), "priority" (prioridade), "completed" (concluída) e "comments" (comentários relacionados à tarefa). A classe Project representa um projeto e possui atributos como "name" (nome do projeto) e "tasks" (array de tarefas associadas a esse projeto).

Essas classes fornecem a estrutura necessária para que a equipe de desenvolvimento possa modelar e implementar as funcionalidades do sistema de gerenciamento de tarefas de acordo com os requisitos definidos. Com base nestas classes, é possível criar instâncias de objetos que representam tarefas e projetos, armazenando as informações necessárias e permitindo a interação e manipulação dos dados de acordo com as necessidades do usuário final.

3.2. Validação e verificação de requisitos convertidos em classe

A validação e verificação garantem que as classes implementadas atendam aos requisitos do sistema e funcionem corretamente. A validação verifica se os requisitos foram interpretados e convertidos corretamente em classes, enquanto a verificação verifica se as classes implementadas estão em conformidade com os requisitos.

Uma técnica importante para a validação e verificação dos requisitos convertidos em classe é a realização de revisões de código. Nesse processo, os membros da equipe revisam o código-fonte das classes para identificar possíveis erros, inconsistências ou violações de boas práticas de programação. Essas revisões garantem a qualidade do código e ajudam a identificar possíveis falhas na implementação dos requisitos.

Outra técnica é a realização de testes. A equipe de desenvolvimento cria casos de teste que verificam o comportamento das classes em relação aos requisitos. Os testes podem abranger diferentes cenários e situações para



garantir que as classes respondam corretamente aos inputs e gerem os outputs esperados. Essa abordagem permite identificar possíveis falhas ou *bugs* nas classes e corrigi-los antes da entrega do produto.



Créditos: Bakhtiar Zein/ Shutterstock.

Nas revisões por pares, os membros da equipe revisam e discutem as classes e sua conformidade com os requisitos. Essa abordagem colaborativa ajuda a identificar possíveis problemas ou lacunas nos requisitos e garante que as classes implementadas estejam alinhadas com as necessidades do cliente.

Essas técnicas de validação e verificação ajudam a identificar possíveis erros, inconsistências ou lacunas nos requisitos, permitindo que a equipe de desenvolvimento faça os ajustes necessários e entregue um produto que atenda às expectativas do cliente. Além disso, essas atividades contribuem para a melhoria contínua do processo de desenvolvimento ágil, garantindo a entrega de um software confiável e de alta qualidade.

TEMA 4 – GERENCIAMENTO DE PROJETOS ÁGEIS

O gerenciamento de projetos ágeis é uma abordagem de gestão que visa facilitar a entrega de projetos de forma adaptativa e colaborativa. Ao contrário



das abordagens tradicionais de gerenciamento de projetos, que seguem planos rígidos e sequenciais, o gerenciamento de projetos ágeis enfatiza a flexibilidade, a interação constante com os clientes e a capacidade de resposta a mudanças.

Nesse contexto, o gerenciamento de projetos ágeis adota uma mentalidade iterativa e incremental, dividindo o projeto em iterações curtas e focando em entregas de valor ao cliente em cada ciclo. Essas iterações são chamadas de "sprints" e, como já vimos anteriormente, geralmente têm uma duração de 1 a 4 semanas. Durante cada sprint, a equipe de projeto se concentra em um conjunto específico de tarefas e objetivos, colaborando de forma intensa e mantendo uma comunicação constante.

Uma das principais metodologias de gerenciamento de projetos ágeis é o Scrum, que define papéis, artefatos e cerimônias específicas para a gestão do projeto. No Scrum, o Product Owner é responsável por definir e priorizar os requisitos do projeto, enquanto a equipe de desenvolvimento é responsável por implementar esses requisitos. O Scrum Master é o facilitador do processo, garantindo que as práticas ágeis sejam seguidas e removendo quaisquer obstáculos que possam surgir.

Outra metodologia ágil comum é o Kanban, que utiliza um quadro visual para rastrear o fluxo de trabalho do projeto. O Kanban permite que a equipe visualize o progresso das tarefas e identifique gargalos ou atrasos rapidamente.

O gerenciamento de projetos ágeis também valoriza a colaboração, o feedback e a melhoria contínuos. A equipe de projeto e os clientes trabalham juntos para refinar os requisitos ao longo do tempo, respondendo a mudanças e ajustando o plano de projeto conforme necessário. A cada iteração, a equipe analisa o trabalho realizado, aprende com as experiências e implementa melhorias para aumentar a eficiência e a qualidade.

4.1 Equipe de projeto ágil

Como vimos no capítulo 3, a equipe de projeto ágil é composta por profissionais multidisciplinares que trabalham de forma colaborativa para entregar o projeto com sucesso. Essa equipe é auto-organizada e autogerenciada, o que significa que ela tem a responsabilidade de planejar, executar e acompanhar o trabalho do projeto.

A equipe de projeto ágil geralmente inclui os seguintes papéis:



1. **Product Owner:** é o representante do cliente ou do usuário final. É responsável por definir as necessidades do cliente, priorizar os requisitos e garantir que o produto atenda às expectativas.
2. **Scrum Master:** é o facilitador do processo ágil. Ajuda a equipe a seguir as práticas e os princípios ágeis, remove os obstáculos que possam surgir e promove a colaboração e a comunicação eficazes.
3. **Equipe de Desenvolvimento:** é composta por profissionais especializados em diferentes áreas, como desenvolvedores, designers, testadores, entre outros. São responsáveis por implementar os requisitos e entregar o produto.



Créditos: LankS/ Shutterstock.

A equipe de projeto ágil realiza reuniões regulares para planejar o trabalho, revisar o progresso, tomar decisões e ajustar o plano conforme necessário. Essa abordagem permite uma maior flexibilidade e adaptabilidade, pois a equipe pode responder rapidamente às mudanças e prioridades do projeto.

A colaboração e a comunicação eficazes são fundamentais para o sucesso da equipe de projeto ágil. A equipe compartilha conhecimentos, troca ideias e busca constantemente melhorar suas práticas e processos. Todos os membros têm voz ativa e contribuem para o sucesso geral do projeto.

4.2 Acompanhamento do que está sendo produzido

No contexto de um projeto ágil, o acompanhamento do que está sendo produzido é realizado de forma contínua e transparente. Existem várias práticas e ferramentas utilizadas para acompanhar o progresso e garantir que as entregas estejam alinhadas com as expectativas e prazos estabelecidos. Alguns dos principais métodos de acompanhamento em projetos ágeis incluem:

1. **Quadro Kanban:** é uma ferramenta visual que permite que a equipe acompanhe o fluxo de trabalho do projeto. As tarefas são representadas por cartões e movidas entre as colunas do quadro, representando o status atual de cada atividade. Isso oferece uma visão clara do que está sendo produzido, o que está em progresso e o que foi concluído.
2. **Burndown Chart:** é uma representação gráfica do trabalho restante ao longo do tempo. Ele mostra a quantidade de trabalho planejada *versus* a quantidade real concluída. Esse gráfico ajuda a equipe a avaliar o progresso do projeto e identificar qualquer desvio em relação ao planejado.



Créditos: shmai/ Shutterstock.

3. **Reuniões diárias (daily stand-ups):** realizadas pela equipe para compartilhar atualizações rápidas sobre o trabalho realizado, os desafios enfrentados e o planejamento para o dia. Essas reuniões permitem que a



equipe acompanhe o progresso, identifique problemas e tome medidas imediatas para resolvê-los.

4. Sprint Review: no final de cada sprint, é realizada uma Sprint Review, na qual a equipe demonstra as entregas realizadas ao Product Owner e aos stakeholders. Essa revisão permite avaliar se as entregas estão atendendo aos requisitos e expectativas definidos e se há necessidade de ajustes para os próximos sprints.

5. Retrospectiva da Sprint: após a conclusão de cada sprint, é realizada uma retrospectiva, na qual a equipe revisa o processo e discute o que funcionou bem e o que pode ser melhorado. Essa reflexão ajuda a identificar oportunidades de melhoria contínua no trabalho e no processo ágil.

Além dessas práticas, outras técnicas, como revisões de código, testes automatizados e feedback contínuo do cliente, também são utilizadas para acompanhar a qualidade e o progresso das entregas.

4.3 Uso de métricas ágeis para medir o progresso do projeto

O uso de métricas ágeis é uma prática fundamental para medir o progresso de um projeto ágil de maneira objetiva e informada. Essas métricas fornecem dados concretos sobre o desempenho da equipe, a qualidade do trabalho entregue e a eficácia do processo ágil. Elas ajudam a equipe a avaliar o progresso em relação aos objetivos do projeto, identificar áreas de melhoria e tomar decisões baseadas em dados.

Algumas métricas mais utilizadas em projetos ágeis incluem:

1. Velocity: é a medida da quantidade de trabalho concluída pela equipe em cada iteração ou sprint. A velocidade é calculada somando-se a quantidade de pontos de história (ou outras unidades de medida) dos itens de backlog concluídos durante a sprint. A velocidade ajuda a equipe a prever o tempo necessário para concluir futuras entregas e a estimar a capacidade de trabalho.

2. Burnup Chart: é um gráfico que mostra o trabalho planejado *versus* o trabalho concluído ao longo do tempo. O eixo vertical representa a quantidade de trabalho, enquanto o eixo horizontal representa o tempo. O burnup chart fornece uma visão clara do progresso do projeto, permitindo



que a equipe compare o trabalho realizado com o planejado e faça ajustes conforme necessário.

3. **Lead Time:** é o tempo necessário para completar um item de backlog, desde o momento em que é iniciado até a sua conclusão. O lead time ajuda a equipe a identificar gargalos no processo, otimizar a eficiência e melhorar a previsibilidade das entregas.
4. **Defeito ou Taxa de Bug:** é a medida da quantidade de defeitos ou erros encontrados no produto ou software entregue. Essa métrica indica a qualidade do trabalho realizado pela equipe e ajuda a identificar áreas que requerem maior atenção ou melhoria.
5. **Retorno sobre o Investimento (ROI):** avalia o valor entregue pelo projeto em relação ao investimento realizado. O ROI mede o benefício financeiro ou estratégico obtido pelo cliente ou organização.

Além dessas métricas, cada projeto ágil pode ter métricas específicas de acordo com seus objetivos e necessidades. É importante escolher métricas relevantes e significativas para o projeto, garantindo que elas sejam consistentemente coletadas e analisadas ao longo do tempo.

4.4 Ajustes e mudanças durante o projeto

Ao contrário das abordagens tradicionais, em que o escopo é fixo, as metodologias ágeis reconhecem que os requisitos e as circunstâncias podem mudar ao longo do tempo. Essa flexibilidade permite que a equipe responda rapidamente a novas informações, necessidades dos usuários e mudanças nas prioridades do negócio. A equipe pode fazer ajustes no planejamento e nas prioridades, garantindo que o projeto permaneça alinhado com os objetivos e entregue valor aos clientes. Essa capacidade de adaptação controlada é uma das principais vantagens das metodologias ágeis, permitindo uma maior eficiência e satisfação do cliente.

4.4.1 Flexibilidade para lidar com mudanças nos requisitos

As metodologias ágeis reconhecem que os requisitos do projeto podem evoluir e mudar ao longo do tempo. Isso pode ocorrer devido a uma série de fatores, como uma melhor compreensão das necessidades dos usuários, novas informações que surgem durante o desenvolvimento ou mudanças nas



prioridades do negócio. Ao invés de considerar as mudanças como algo negativo, as metodologias ágeis encorajam a adaptação e a flexibilidade para atender a essas mudanças de maneira eficiente.

À medida que o projeto progride e a equipe ganha mais informações, novas histórias de usuário podem ser adicionadas ao backlog ou as existentes podem ser alteradas. A flexibilidade nas metodologias ágeis permite que essas mudanças sejam incorporadas sem causar grandes interrupções no fluxo de trabalho. A equipe pode priorizar as histórias de usuário de acordo com o valor que elas agregam ao produto e, assim, fazer escolhas informadas sobre quais histórias abordar em cada sprint.

4.4.2 Adaptação do planejamento e prioridades

As equipes ágeis desenvolvem um plano inicial com base nos requisitos e nas informações disponíveis no momento. Esse plano inicial serve como uma linha de base, mas é esperado que ele seja adaptado ao longo do tempo.

À medida que a equipe avança no desenvolvimento do projeto e ganha mais conhecimento, novas informações podem surgir. Essas informações podem indicar que certas tarefas são mais complexas do que o esperado, ou que certos requisitos precisam ser ajustados para melhor atender às necessidades dos usuários. A equipe ágil tem a capacidade de revisar o planejamento inicial e fazer ajustes com base nessas informações atualizadas.

Da mesma forma, as prioridades também podem mudar ao longo do projeto. Novas demandas podem surgir ou a equipe pode perceber que certas funcionalidades são mais importantes do que outras. A capacidade de adaptar as prioridades permite que a equipe concentre seus esforços nas áreas que trarão maior valor ao produto.

Para gerenciar essas mudanças de maneira estruturada, as metodologias ágeis possuem cerimônias e práticas específicas. Por exemplo, a revisão do backlog pode ser realizada regularmente para reavaliar e repriorizar as histórias de usuário. Além disso, reuniões diárias, conhecidas como daily stand-ups, permitem que a equipe compartilhe atualizações, identifique possíveis obstáculos e ajuste o curso do projeto conforme necessário.

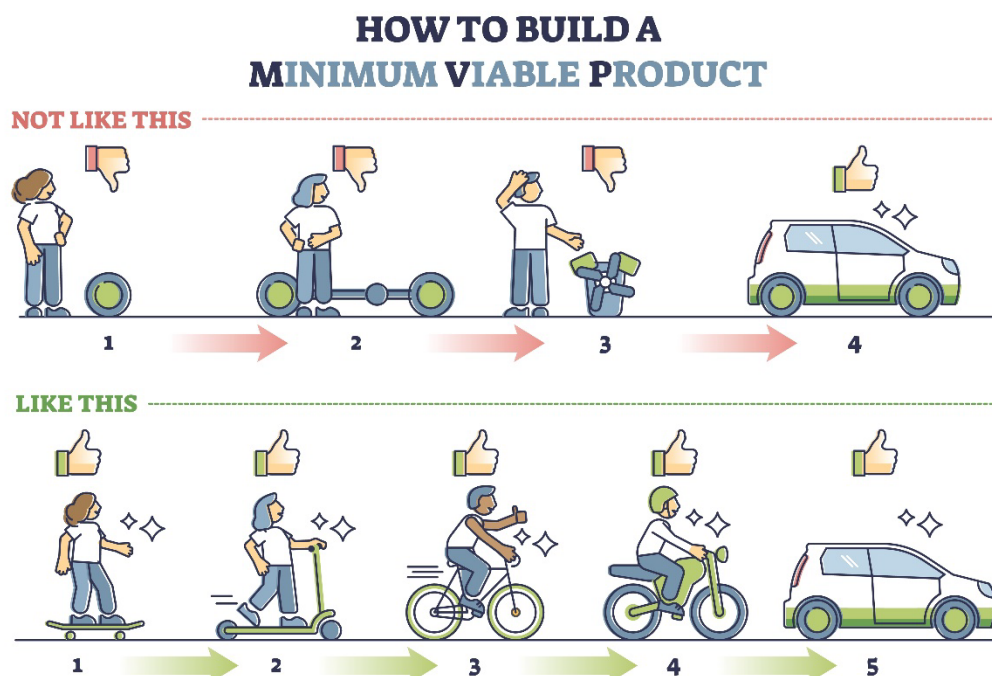
Essa abordagem permite que as equipes ágeis respondam de maneira mais rápida e eficiente às mudanças, minimizando riscos e maximizando o valor entregue ao cliente. Ao longo do projeto, a equipe aprende com as iterações

anteriores, refina o planejamento e melhora a capacidade de lidar com mudanças e incertezas.

4.5 MVP - Mínimo Produto Viável

O MVP é uma prática que consiste em criar e lançar um produto com o mínimo de recursos e funcionalidades necessários para atender às necessidades do cliente e validar a proposta de valor. O objetivo principal do MVP é obter feedback e aprendizado rápido por meio da interação com os usuários, permitindo que a equipe ajuste e aprimore o produto com base nas necessidades reais do mercado.

A importância do MVP reside no fato de que ele evita o desperdício de tempo e recursos no desenvolvimento de funcionalidades que podem não ser relevantes ou valorizadas pelos usuários. Ao lançar uma versão inicial do produto, a equipe pode testar suas hipóteses e validar suas ideias de forma rápida e econômica. Além disso, o MVP ajuda a reduzir os riscos associados ao desenvolvimento, fornecendo *insights* sobre a aceitação e o desempenho do produto no mercado.



Créditos: VectorMine/ Shutterstock.



4.5.1 Definição do escopo mínimo para entrega de valor

Ao desenvolver um MVP, é essencial definir o escopo mínimo necessário para entregar valor aos usuários. Isso significa identificar as funcionalidades-chave que são essenciais para atender às necessidades iniciais dos clientes. Essa abordagem evita a sobrecarga de recursos e permite que a equipe se concentre no desenvolvimento das partes mais importantes do produto.

A definição do escopo mínimo também envolve o estabelecimento de critérios claros para determinar o que é necessário para lançar o MVP. Esses critérios podem incluir a estabilidade do produto, a funcionalidade mínima requerida e a experiência do usuário básica. Ao estabelecer esses critérios, a equipe pode garantir que o MVP cumpra os requisitos mínimos para ser lançado e testado.

4.5.2 Iterações sucessivas para aprimoramento do produto

Uma vez lançado o MVP, a equipe inicia um processo de iterações sucessivas para aprimorar o produto com base no feedback dos usuários. Essas iterações, geralmente realizadas em sprints, permitem que a equipe colete informações valiosas sobre o uso do produto, identifique áreas de melhoria e implemente ajustes e adições incrementais.

O ciclo de feedback contínuo nas metodologias ágeis possibilita que o produto evolua de forma ágil e iterativa. A equipe analisa os resultados do MVP, faz ajustes, adiciona novas funcionalidades e reprioriza o trabalho com base nas necessidades dos usuários e nas mudanças do mercado. Esse processo iterativo de desenvolvimento permite uma melhoria contínua do produto, garantindo que ele se alinhe cada vez mais às expectativas e necessidades dos clientes.

TEMA 5 - CONFIGURAÇÃO DE PROJETOS ÁGEIS

A configuração de um projeto ágil estabelece as bases sólidas necessárias para que as metodologias ágeis sejam implementadas de forma eficiente e eficaz.



Créditos: ankastudio22/ Shutterstock.

Aqui estão algumas razões pelas quais a configuração adequada é fundamental:

- **Alinhamento estratégico:** permite alinhar o trabalho da equipe com os objetivos estratégicos da organização, define a visão do projeto, os resultados desejados e as metas a serem alcançadas, garantindo que o trabalho esteja alinhado com as necessidades e prioridades da empresa.
- **Definição clara de papéis e responsabilidades:** a configuração adequada estabelece os papéis e responsabilidades de cada membro da equipe. Isso evita ambiguidade e confusão, permitindo que todos entendam suas funções e contribuições para o projeto. Dessa forma, a colaboração é facilitada e as decisões podem ser tomadas de forma mais eficiente.
- **Comunicação eficaz:** uma configuração bem estruturada inclui práticas de comunicação claras e eficazes, garantindo que todos os membros da equipe estejam informados sobre o progresso do projeto, desafios enfrentados e próximos passos.
- **Flexibilidade para lidar com mudanças:** permite que os projetos sejam flexíveis e se adaptem às mudanças inevitáveis. Fornece estruturas e processos que permitem uma resposta rápida às alterações de requisitos, prioridades e circunstâncias.
- **Entrega contínua de valor:** facilita a entrega contínua de valor ao cliente. Ao estabelecer iterações curtas e incrementais, permite que a equipe



entregue funcionalidades utilizáveis e testáveis em um curto espaço de tempo.

- **Melhoria contínua:** incentiva a melhoria contínua ao longo do projeto. Por meio de práticas como retrospectivas, a equipe pode identificar oportunidades de aprimoramento, ajustar processos e adotar abordagens mais eficazes.

Supondo que estamos configurando um projeto ágil de desenvolvimento de um aplicativo móvel, aqui está um exemplo prático de como seria a configuração:

1. Infraestrutura e ambientes de desenvolvimento:

- Configuração de um servidor de controle de versão, como o Git, para gerenciar o código-fonte do aplicativo;
- Utilização de uma plataforma de hospedagem de repositórios, como o GitHub, para facilitar o compartilhamento e a colaboração entre os desenvolvedores;
- Escolha de uma ferramenta de gerenciamento de projetos, como o Jira, para acompanhar as tarefas, atribuir responsabilidades e monitorar o progresso do projeto;
- Implementação de ferramentas de comunicação, como o Slack, para facilitar a comunicação instantânea e a colaboração entre os membros da equipe.

2. Equipe multifuncional:

- Formação de uma equipe composta por desenvolvedores, designers de interface do usuário (UI) e designers de experiência do usuário (UX), permitindo uma abordagem multidisciplinar no desenvolvimento do aplicativo;
- Definição clara de papéis e responsabilidades, com um Product Owner para definir as necessidades dos usuários, um Scrum Master para facilitar o processo ágil e a equipe de desenvolvimento para implementar as funcionalidades do aplicativo.

3. Ambientes de testes e integração contínua:

- Configuração de ambientes de teste separados para testes funcionais, de regressão e de desempenho do aplicativo;



- Utilização de ferramentas de automação de testes para realizar testes de interface do usuário de forma eficiente;
- Implementação de práticas de integração contínua para integrar, compilar e testar automaticamente o código em um ambiente compartilhado.

4. Iterações e entregas frequentes:

- Definição de sprints, períodos curtos (por exemplo, duas semanas), nos quais a equipe se concentra em entregar um conjunto de funcionalidades prioritárias;
- Realização de reuniões diárias (daily stand-ups) para alinhar as atividades e discutir eventuais obstáculos;
- Realização de revisões de sprint, nas quais a equipe demonstra as funcionalidades concluídas e coleta feedback dos stakeholders;
- Aproveitamento do feedback recebido para ajustar as prioridades e a direção do projeto em iterações subsequentes.

Essas são apenas algumas das configurações práticas que podem ser implementadas em um projeto ágil. É importante adaptar a configuração de acordo com as necessidades e particularidades do projeto e da equipe. A configuração deve ser flexível o suficiente para se adaptar a mudanças e permitir a evolução contínua do produto.

FINALIZANDO

Nesta abordagem, vimos diversos tópicos relacionados à gestão ágil de projetos. No primeiro deles, exploramos o Planejamento do Projeto, destacando a metodologia *Agile Planning Onion*. Em seguida, discutimos Requisitos em projetos ágeis, apresentando exemplos e técnicas para o levantamento ágil de requisitos.

No terceiro tópico, focamos na Conversão de Requisitos em Classe, exemplificando o processo e ressaltando a importância da validação e verificação dos requisitos convertidos. O quarto tópico tratou do Gerenciamento de Projetos Ágeis, abordando a formação da equipe, o acompanhamento do trabalho em andamento e o uso de métricas ágeis para medir o progresso do projeto. Também enfatizamos a flexibilidade para lidar com mudanças nos requisitos e a adaptação do planejamento e prioridades.



Por fim, no tópico 5, discutimos a Configuração de Projetos Ágeis, destacando a importância de configurar corretamente o ambiente de trabalho e a infraestrutura necessária para o desenvolvimento ágil. Ao longo desta abordagem, também vimos o conceito de MVP (Mínimo Produto Viável), que consiste em definir um escopo mínimo para a entrega de valor e realizar iterações sucessivas para aprimorar o produto.



REFERÊNCIAS

GOMES, A. F. **Agile**: desenvolvimento de software com entregas frequentes e foco no valor de negócio. São Paulo: Casa do Código, 2013.

JAQUEIRA, A. et al. Desafios de requisitos em Métodos Ágeis: uma revisão sistemática. In: Workshop Brasileiro de Métodos Ágeis, 3, 2012, São Paulo. **Anais**. São Paulo: USP, 2012.

KÄPYAHO, M.; KAUPPINEN, M. Agile requirements engineering with prototyping: A case study. **International Requirements Engineering Conference**, 2015.

SBROCCO, J. H. T. de C.; MACEDO, P. C. **Metodologias ágeis**: engenharia de software sob medida. São Paulo: Editora Érica, 2012.

SOMMERVILLE, I. **Engenharia de software**. 8. ed. São Paulo: Pearson Addison Wesley, 2007.