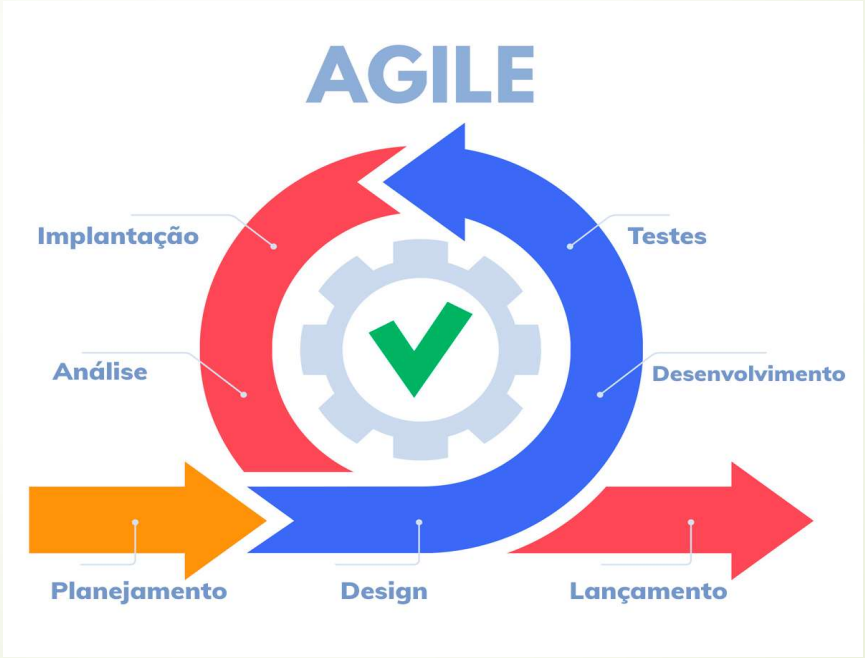
FACULDADE DE CIÊNCIAS E TECNOLOGIAS DE INFORMAÇÃO

ADMINISTRAÇÃO DE SISTEMAS DE IMFORMAÇÃO E REDES

Introduçao

- O modelo Ágil é uma abordagem dinâmica e iterativa para o desenvolvimento de software, que privilegia a entrega célere e contínua de valor ao cliente.
- o Ágil divide o trabalho em ciclos curtos chamados sprints, onde o feedback constante do cliente é incorporado para adaptar e melhorar o produto ao longo do tempo
- Esta metodologia coloca um forte ênfase nas pessoas e nas interações, na colaboração próxima com o cliente, e na capacidade de adaptação rápida às mudanças, tornando-se ideal para projetos em que os requisitos estão em constante evolução.


AGILE
Implantação
Testes
Análise
Desenvolvimento
Planejamento
Design
Lançamento

Modelo de Desenvolvimento de software enxuto (LEAN)

- **Origem do Desenvolvimento de Software Enxuto**
- O Desenvolvimento Enxuto (Lean) tem suas raízes na **Lean Manufacturing**, uma abordagem de produção desenvolvida pela Toyota na década de 1950, conhecida como **Sistema Toyota de Produção (TPS)**. A principal filosofia do TPS era eliminar desperdícios e aumentar a eficiência no processo de produção. Mary e Tom Poppendieck adaptaram esses princípios para o desenvolvimento de software em seu livro "Lean Software Development: An Agile Toolkit", publicado em 2003.
- O Desenvolvimento Enxuto de Software, também conhecido como Lean Software Development, é uma abordagem que visa otimizar o processo de desenvolvimento, tornando-o mais eficiente e eliminando desperdícios.

1. Eliminar Desperdícios:

•**Desperdício** é tudo o que não agrega valor ao cliente. No desenvolvimento de software,
•isso pode incluir funcionalidades que não são usadas, código duplicado, burocracia desnecessária, entre outros.
•**Exemplos de desperdício** a serem eliminados:

- **Requisitos não claros ou mal definidos.**
- **Funções não utilizadas ou desnecessárias.**
- **Processos de aprovação complexos.**
- **Defeitos e retrabalho.**

2. Amplificar o Aprendizado:

•**Ciclos curtos de feedback** ajudam a equipe a aprender rapidamente o que funciona e o que não funciona.
•**Prototipagem, testes contínuos** e **revisões de código** são práticas comuns para fomentar o aprendizado.
•A documentação deve ser mínima, mas suficiente para transmitir conhecimento essencial, evitando a burocracia.

Princípios Fundamentais (cont)

- **3. Decisão Tardia (Decidir o Mais Tarde Possível):**
- **Decisões importantes** devem ser adiadas até que se tenha informações suficientes para tomar a melhor decisão.
- Isso aumenta a flexibilidade do projeto, permitindo que as equipes se adaptem melhor a mudanças de requisitos ou novas tecnologias.

4. Entrega Rápida:
•A ênfase está em entregar software que funcione rapidamente e frequentemente, para que o cliente possa começar a usar o produto e fornecer feedback.
•**MVP (Produto Mínimo Viável)** é uma prática comum, onde uma versão funcional e mínima do produto é lançada o mais rápido possível.

5. Empoderar a Equipe:
•As decisões são tomadas o mais próximo possível da equipe que está executando o trabalho.
•**Autonomia** e **responsabilidade** são incentivadas, e a liderança deve ser servidora, facilitando o trabalho da equipe em vez de controlá-lo rigidamente.

• **6. Construir Qualidade desde o Início:**
A qualidade não deve ser algo que se verifica apenas no final do processo, mas deve ser integrada em cada fase do desenvolvimento.
•**Práticas como TDD (Test Driven Development)**, **integração contínua** e **pair programming** ajudam a garantir que o software seja de alta qualidade desde o início.

• **7. Otimizar o Todo:**
•Em vez de focar em otimizar partes isoladas do processo (como desenvolvimento ou testes), o Lean busca otimizar o sistema como um todo.
•A visão holística garante que as melhorias em uma parte do processo não criem gargalos ou problemas em outra.

Práticas Comuns

•**MVP (Produto Mínimo Viável)**: Lançar versões básicas e funcionais do produto para obter feedback precoce.

•**Kanban**: Visualização do fluxo de trabalho para gerenciar o trabalho em progresso (WIP).

•**Automação de Testes**: Garantir que o software seja continuamente testado e mantido em alta qualidade.

• **Integração Contínua (CI):** Integração frequente de código para detectar problemas rapidamente.

•**Desenvolvimento Orientado a Testes (TDD):** Escrita de testes antes do código funcional para garantir que cada unidade do software atenda aos requisitos.

Vantagens do Lean Software Development

- **Eficiência**: Redução de desperdícios, tornando o processo mais ágil e focado.
- **Qualidade**: A ênfase em práticas de qualidade desde o início resulta em software mais robusto e confiável.
- **Flexibilidade**: Adaptação rápida às mudanças, graças à tomada de decisões tardias e ao feedback constante.
- **Satisfação do Cliente:** A entrega contínua de valor e o foco nas necessidades reais do cliente aumentam a satisfação e a confiança no processo de desenvolvimento.

Desafios

- **Mudança Cultural**: Requer uma mudança significativa na cultura organizacional para adotar os princípios Lean.
- **Necessidade de Equipes Experientes**: O sucesso depende da competência e da experiência da equipe.
- **Complexidade na Otimização do Todo**: Exige uma visão holística para otimizar o sistema completo, especialmente em grandes organizações.

Desvantagens e Desafios do Desenvolvimento de Software Enxuto

- **Mudança Cultural e Resistência:**
  - **Desafio:** Necessidade de mudar a cultura organizacional pode enfrentar resistência.
  - **Solução:** Investir em comunicação clara e treinamento para facilitar a transição
- **Necessidade de Equipes Altamente Experientes:**
  - **Desafio:** Equipes inexperientes podem ter dificuldade em aplicar os princípios Lean.
  - **Solução:** Proporcionar treinamento e desenvolvimento contínuo para a equipe.
- **Complexidade na Otimização do Todo:**
  - **Desafio:** Otimizar o sistema completo pode ser complexo, especialmente em grandes organizações.
  - **Solução:** Utilizar frameworks de escalabilidade e práticas de gerenciamento de portfólio.

Desvantagens e Desafios do Desenvolvimento de Software Enxuto (cont)

- **Foco Excessivo em Processos e Ferramentas:**
- **Desafio:** Pode haver uma ênfase excessiva em processos em detrimento das necessidades do cliente.
- **Solução:** Manter o foco na entrega de valor ao cliente e aplicar os princípios Lean com flexibilidade.
- **Risco de Subotimização:**
- **Desafio:** A eliminação de desperdícios pode levar à subotimização, prejudicando a eficiência global.
- **Solução:** Avaliar a eficiência global usando métricas como Lead Time e Cycle Time.
- **Desafios em Organizações Grandes:**
- **Desafio:** Implementar Lean em grandes organizações pode ser complicado devido à complexidade.
- **Solução:** Iniciar com projetos piloto e expandir gradualmente

Lean e Scrum

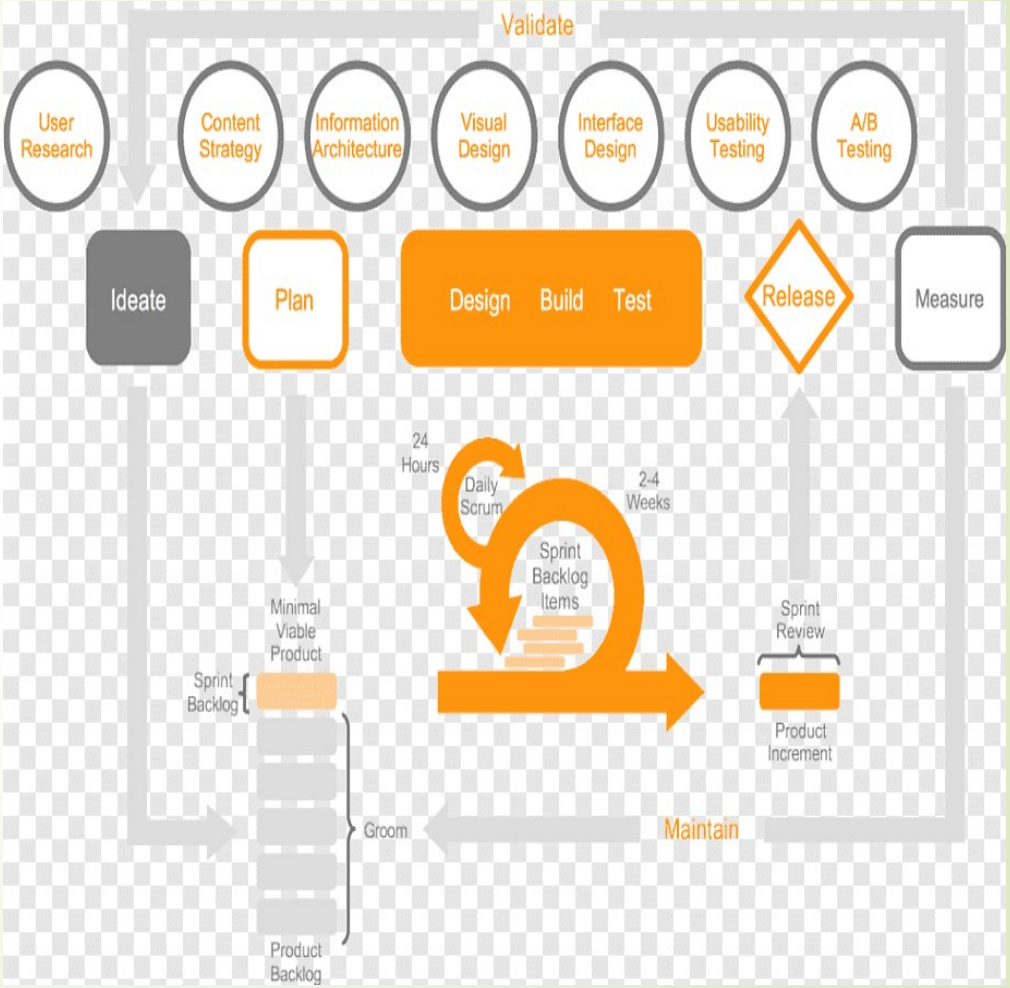
- **Integração de Princípios:**
- **Lean com Scrum:** Lean pode ser integrado ao Scrum para melhorar a eficiência e eliminar desperdícios durante o processo de Sprints. Por exemplo, Kanban pode ser usado dentro de um framework Scrum para gerenciar o fluxo de trabalho e visualização do progresso.
- **Benefícios da Combinação:**
- **Eficiência e Agilidade:** Lean ajuda a identificar e eliminar desperdícios, enquanto Scrum fornece uma estrutura para organizar e gerenciar o trabalho de forma iterativa e colaborativa.
- **Melhor Visibilidade e Controle:** Kanban pode melhorar a visibilidade do trabalho dentro dos Sprints, e práticas Lean podem otimizar a forma como o trabalho é priorizado e executado.

- **Desafios da Integração:**
- **Complexidade:** Integrar Lean e Scrum pode adicionar complexidade, especialmente em termos de gerenciamento e coordenação dos processos.
- **Adaptação de Práticas:** Pode ser necessário adaptar práticas Lean e Scrum para garantir que elas se complementem eficazmente.

Siglas

- **Test Driven Development (TDD)**, ou **Desenvolvimento Orientado por Testes**, é uma metodologia de desenvolvimento de software na qual os testes são escritos antes do código que será testado.
- **Pair Programming** (ou **Programação em Par**) é uma prática de desenvolvimento de software onde dois programadores trabalham juntos em uma única estação de trabalho. Um dos programadores, chamado de "piloto", escreve o código, enquanto o outro, chamado de "navegador", revisa cada linha de código à medida que é escrita. Eles trocam de papéis regularmente.
- **Kanban** é uma metodologia de gerenciamento de trabalho que visa melhorar a eficiência e a produtividade, visualizando o fluxo de trabalho e limitando o trabalho em progresso (WIP).
- **Lead Time** é o tempo total que se passa desde o momento em que uma tarefa ou item de trabalho é solicitado até o momento em que é entregue e está pronto para uso. Em outras palavras, é o tempo que um cliente ou solicitante espera para que a tarefa seja concluída.
- **Cycle Time** é o tempo que uma tarefa ou item de trabalho leva para ser concluído desde o momento em que começa a ser trabalhado até o momento em que é finalizado.

- **1. Ideate (Ideação)**: Geração de ideias e definição de requisitos iniciais.
- **2. Plan (Planejamento)**: Planejamento das funcionalidades e tarefas a serem desenvolvidas.
- **3. Design (Design)**: Criação do design do software, incluindo interfaces e arquitetura.
- **4. Build (Construção)**: Desenvolvimento do código e implementação das funcionalidades.
- **5. Test (Teste)**: Testes para garantir que o software funcione corretamente.
- **6. Release (Lançamento)**: Lançamento da versão do software para os usuários.
- **7. Maintain (Manutenção)**: Manutenção contínua e melhorias do software.



Conclusão

•**Resumo**: O Desenvolvimento de Software Enxuto é uma abordagem poderosa que combina eficiência, qualidade e flexibilidade.

•**Aplicabilidade**: Ideal para organizações que desejam uma entrega contínua de valor ao cliente, com a capacidade de se adaptar rapidamente às mudanças do mercado e dos requisitos.