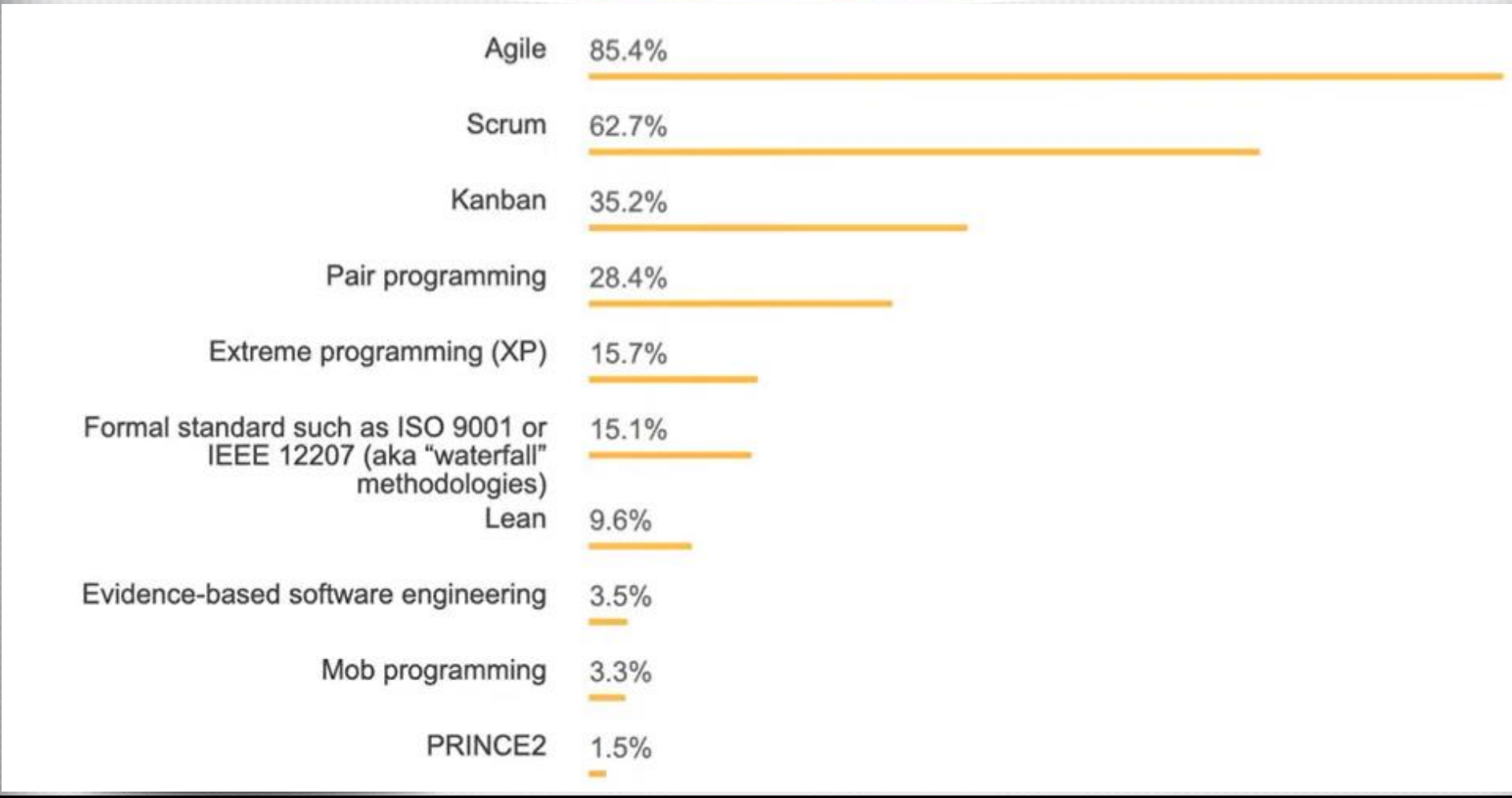
SENAI

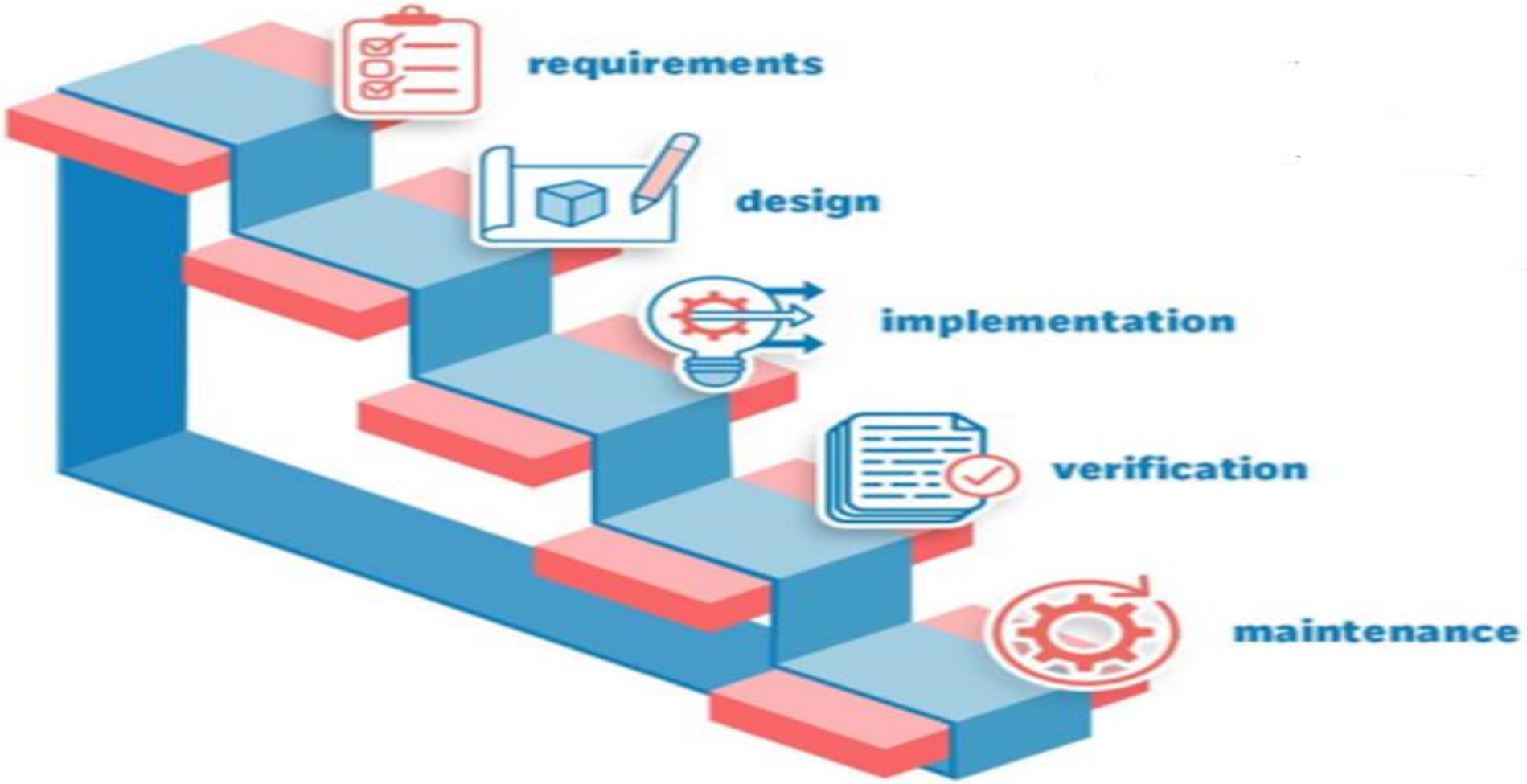
Faculdade de Tecnologia SENAI "Roberto Mange"

Engenharia de Software
Processos

Utilização de metodologias por desenvolvedores 2018



Método Waterfall (Cascata)



Método Waterfall (Cascata)

1. ***Introdução à Metodologia Waterfall***:

- **Definição**: Descreva o que é a metodologia Waterfall e como funciona.
- **Histórico:** Apresente a origem e contexto histórico da metodologia Waterfall.
- ***Princípios Fundamentais***: Explique os principais princípios que regem a metodologia Waterfall, como linearidade, planejamento detalhado e sequencialidade.

2. Fases da Metodologia Waterfall:

- **Análise de Requisitos**: Descreva detalhadamente esta fase, incluindo:
  - **Objetivo:** Definir o que o sistema deve fazer e como deve se comportar.
  - **Técnicas**: Abordar técnicas como entrevistas, questionários, análise de documentos, etc.
  - **Documentação**: Explicar a importância da documentação dos requisitos (especificação de requisitos).

Método Waterfall (Cascata)

- ***Projeto:***
  - **Objetivo:** Traduzir os requisitos em um projeto técnico detalhado, incluindo arquitetura, design de interfaces, banco de dados, etc.
  - **Técnicas:** Abordar técnicas como diagramação UML, modelos de dados, etc. Documentação: Explicar a importância da documentação do projeto (especificação de design).

- ***Implementação:***
  - ***Objetivo:*** Traduzir o projeto em código executável.
  - **Técnicas:** Abordar linguagens de programação, frameworks, ferramentas de desenvolvimento, etc.
  - **Documentação:** Explicar a importância da documentação do código (documentação de código).

Método Waterfall (Cascata)

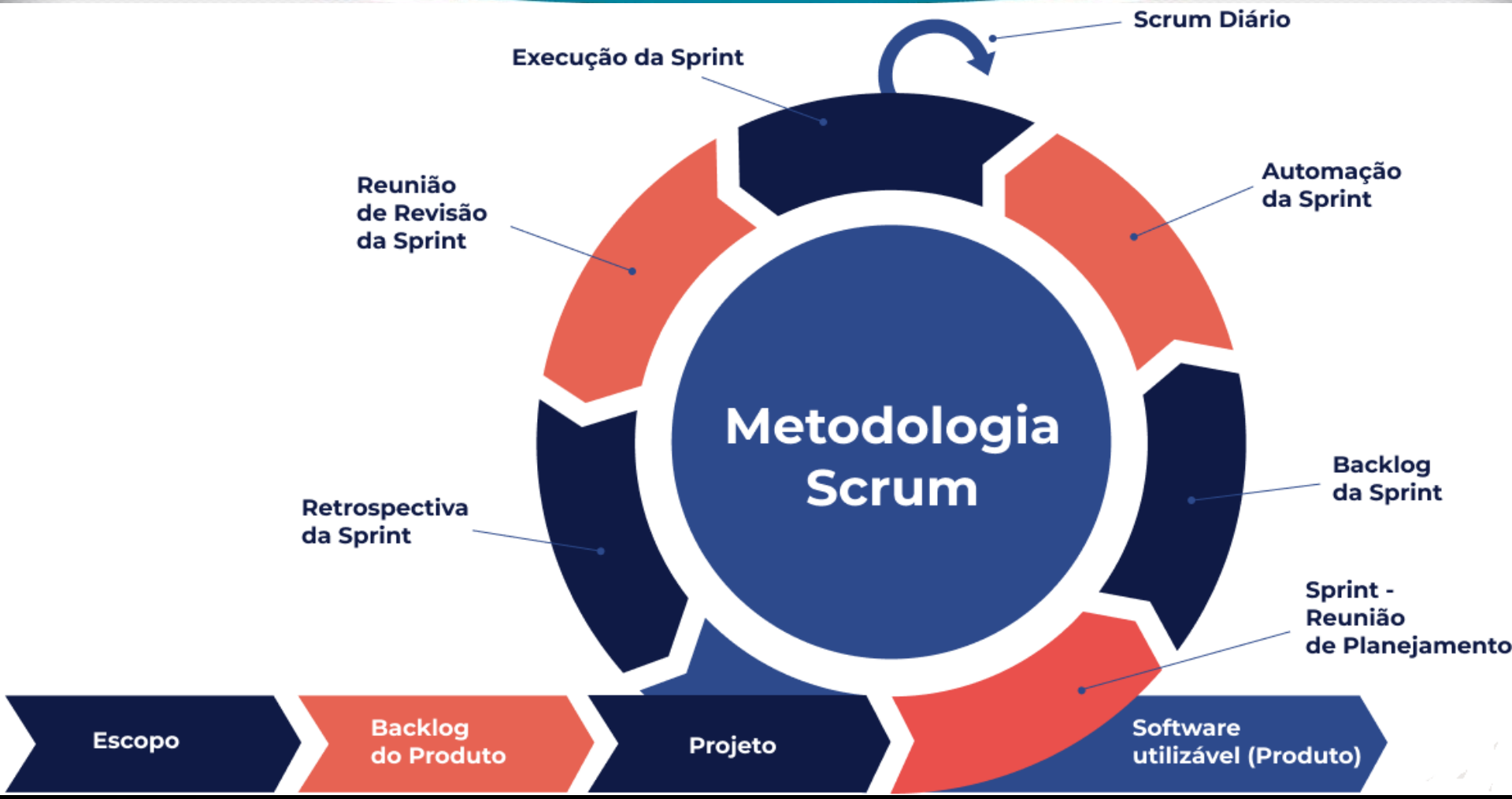
- ***Teste:***
  - ***Objetivo:*** Verificar se o sistema implementa os requisitos e funciona corretamente.
  - ***Tipos de Testes:*** Abordar diferentes tipos de testes (unitários, de integração, de sistema, de aceitação, etc.).
  - ***Documentação:*** Explicar a importância da documentação dos testes realizados (relatórios de testes).
- ***Implantação:***
  - ***Objetivo:*** Instalar e configurar o sistema no ambiente de produção.
  - ***Aspectos:*** Abordar a instalação, configuração, treinamento de usuários, migração de dados, etc.
  - ***Documentação:*** Explicar a importância da documentação do processo de implantação (manual de implantação).

Método Waterfall (Cascata)

- ***Manutenção:***
  - ***Objetivo:*** Corrigir erros, realizar atualizações e aprimoramentos no sistema após a implantação.
  - ***Tipos de Manutenção:*** Abordar os diferentes tipos de manutenção (corretiva, adaptativa, preventiva, etc.).
  - ***Documentação:*** Explicar a importância da documentação das atividades de manutenção (relatórios de manutenção).

O seminário deve ser claro, conciso e objetivo, com informações precisas e exemplos relevantes para ilustrar os conceitos abordados.

Método Ágil ➔ SCRUM



Método Ágil ➔ SCRUM

1. ***Introdução ao Scrum***

- ***Contextualização:*** A crise do desenvolvimento de software tradicional e a necessidade de métodos ágeis.
- ***Definição:*** O que é Scrum? É um framework ágil para gerenciamento de projetos.
- ***Princípios do Scrum:***
  - ***Transparência:*** Visualização constante do progresso e dos desafios.
  - ***Inspeção:*** Avaliação frequente e crítica do processo e do produto.
  - ***Adaptação:*** Flexibilidade para ajustar o plano e o produto em resposta ao feedback.
- ***Valores do Scrum:*** Compromisso, Foco, Abertura, Respeito e Coragem.

Método Ágil ➔ SCRUM

2. As Pessoas no Scrum:

•***Papéis:***

- ***Product Owner:*** Responsável pela visão do produto, prioriza o backlog e valida as funcionalidades.
- ***Scrum Master:*** Facilitador do processo Scrum, garante a aplicação das práticas e remove impedimentos.
- ***Development Team:*** Responsável por desenvolver e testar o produto.

•***Importância da auto-organização:*** O time decide como realizar as tarefas.

•***Comunicação e colaboração:*** Exercício prático: simule uma equipe Scrum e a interação entre os papéis.

Método Ágil ➔ SCRUM

3. Artefatos do Scrum:

- ***Product Backlog:*** Lista priorizada de funcionalidades do produto, representada por "user stories" detalhadas.
- ***Sprint Backlog:*** Conjunto de tarefas que o time se compromete a entregar em uma Sprint.
- ***Increment:*** Produto funcional e incremental gerado no final de cada Sprint.
- ***Burn Down Chart:*** Visualização gráfica do progresso do time durante a Sprint.
- ***Demonstração:*** Apresentação do Increment ao Product Owner e stakeholders

Método Ágil ➔ SCRUM

4. Cerimônias do Scrum:

- ***Sprint Planning:*** Planejamento detalhado do trabalho a ser realizado durante a Sprint.
- ***Daily Scrum:*** Reunião diária para atualização do progresso, identificação de impedimentos e alinhamento do time.
- ***Sprint Review:*** Apresentação do Increment e coleta de feedback.
- ***Sprint Retrospective:*** Reunião para análise e melhoria do processo Scrum.
- ***Exercício prático:*** Simulação de uma Sprint Planning e Daily Scrum.

Método Ágil ➔ SCRUM

5. Aplicações Práticas do Scrum em Desenvolvimento de Software:

- ***Desenvolvimento de um sistema web:*** Exemplificação com user stories, sprints e entrega incremental.
- ***Desenvolvimento de um aplicativo mobile:*** Apresentação das funcionalidades, etapas e iterações

6. Benefícios do Scrum:

- ***Flexibilidade:*** Adaptação a mudanças e entrega de valor incremental.
- ***Comunicação e colaboração:*** Melhora na comunicação e colaboração entre os membros da equipe.
- ***Qualidade e entrega rápida:*** Prioriza a entrega de valor e qualidade.
- ***Transparência e feedback:*** Promove a transparência e o feedback contínuo.

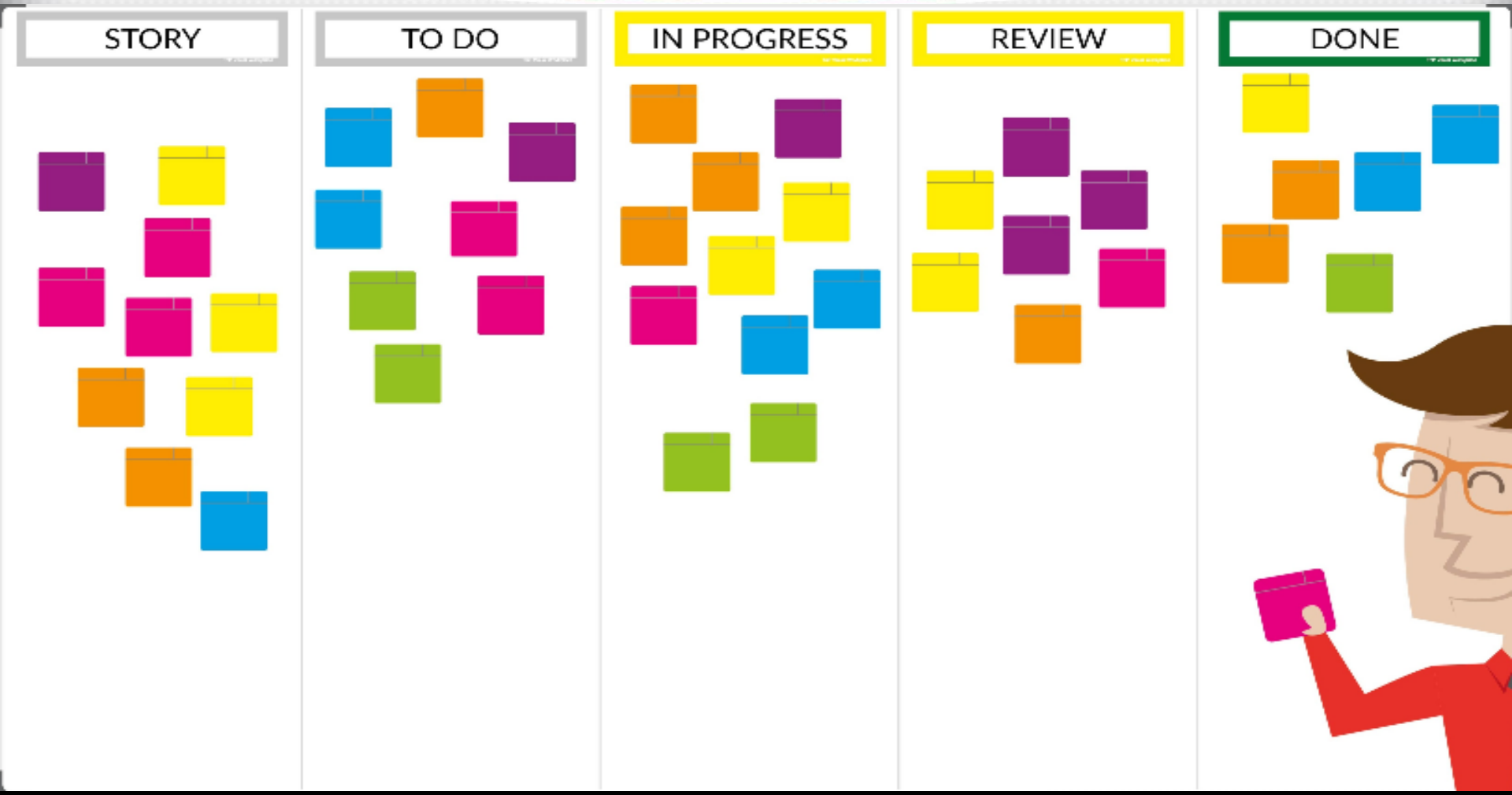
Método Ágil ➔ SCRUM

7. Desafios e Dicas para a Implementação do Scrum:

- ***Resistência à mudança:*** Como lidar com a resistência à implementação do Scrum.
- ***Cultura organizacional:*** Adaptação do Scrum à cultura da empresa.
- ***Treinamento e mentoria:*** Importância do treinamento e mentoria para o sucesso do Scrum.
- ***Escolha da ferramenta:*** Opções de ferramentas para gestão de projetos Scrum.

O seminário deve ser claro, conciso e objetivo, com informações precisas e exemplos relevantes para ilustrar os conceitos abordados.

Método Ágil ➔ KANBAN



Método Ágil ➔ KANBAN

1. Introdução ao Kanban:

- ***Contextualização:*** A necessidade de métodos ágeis para lidar com a complexidade e a mudança constante no desenvolvimento de software.
- ***Definição:*** O que é Kanban? É um método visual para gerenciar o fluxo de trabalho, focando na otimização e na entrega contínua de valor.
- ***Princípios do Kanban:***
  - ***Visualização do fluxo de trabalho:*** Representar o trabalho em andamento de forma clara e transparente.
  - ***Limitação do trabalho em andamento (WIP):*** Controlar o número de tarefas em andamento para evitar gargalos e atrasos.
  - ***Fluxo contínuo:*** Manter o fluxo de trabalho constante e identificar pontos de bloqueio.
  - ***Melhoria contínua:*** Buscar aperfeiçoamento constante do processo através de feedback e iterações.

Método Ágil ➔ KANBAN

2. Implementação do Kanban:

- ***Criação do Kanban Board:*** Visualizar as etapas do fluxo de trabalho através de colunas (ex.: "A Fazer", "Em Andamento", "Concluído").
- ***Cartões Kanban:*** Representar cada tarefa através de cartões com informações relevantes (ex.: título, descrição, prazo, responsável).
- ***WIP Limits:*** Definir o número máximo de tarefas permitidas em cada etapa do processo.
- ***Metricas Kanban:*** Monitorar o desempenho do fluxo de trabalho com métricas como Lead Time (tempo total de entrega) e Cycle Time (tempo de entrega de cada tarefa).

Método Ágil ➔ KANBAN

3. Práticas do Kanban:

- ***Pull System:*** As tarefas são puxadas para a próxima etapa do fluxo de trabalho somente quando a etapa anterior estiver pronta.
- ***Priorização:*** Definir prioridades para as tarefas, utilizando um sistema de classificação (ex.: alta, média, baixa).
- ***Gestão de Impedimentos:*** Identificar e remover os obstáculos que impedem o fluxo de trabalho.
- ***Feedback e Retrospectivas:*** Avaliar e melhorar continuamente o processo de trabalho.

Método Ágil ➔ KANBAN

4. Aplicações Práticas do Kanban em Desenvolvimento de Software:

- ***Gerenciamento de tarefas de desenvolvimento:*** Utilizar o Kanban Board para organizar as atividades do time.
- ***Gerenciamento de bugs:*** Implementar um fluxo Kanban para rastrear e resolver bugs.
- ***Gerenciamento de demandas:*** Utilizar o Kanban para priorizar e organizar as demandas dos clientes.
- ***Integração com outras ferramentas:*** Combinar o Kanban com ferramentas de gestão de projetos, versionamento de código e comunicação.

Método Ágil ➔ KANBAN

5. Benefícios do Kanban:

- ***Melhor visibilidade do fluxo de trabalho:*** Facilidade para identificar gargalos e atrasos.
- ***Redução do tempo de entrega:*** Otimizar o fluxo de trabalho e diminuir o tempo de entrega das tarefas.
- ***Melhoria da comunicação e colaboração:*** Compartilhar informações e promover a colaboração entre as equipes.
- ***Adaptação a mudanças:*** Flexibilidade para adaptar o fluxo de trabalho às novas demandas.
- ***Foco na entrega de valor:*** Priorizar as tarefas que agregam mais valor ao cliente.

Método Ágil ➔ KANBAN

6. Desafios e Dicas para a Implementação do Kanban:

- ***Resistência à mudança:*** Como lidar com a resistência à implementação do Kanban.
- ***Cultura organizacional:*** Adaptação do Kanban à cultura da empresa.
- ***Treinamento e mentoria:*** Importância do treinamento e mentoria para o sucesso do Kanban.
- ***Escolha da ferramenta:*** Opções de ferramentas para gestão de Kanban (ex.: Trello, Jira, Asana).

7. Comparação entre Kanban e Scrum:

- ***Diferenças:*** Apresentar as principais diferenças entre os métodos Kanban e Scrum.
- ***Combinação:*** Como combinar os métodos Kanban e Scrum para obter os melhores resultados***.***

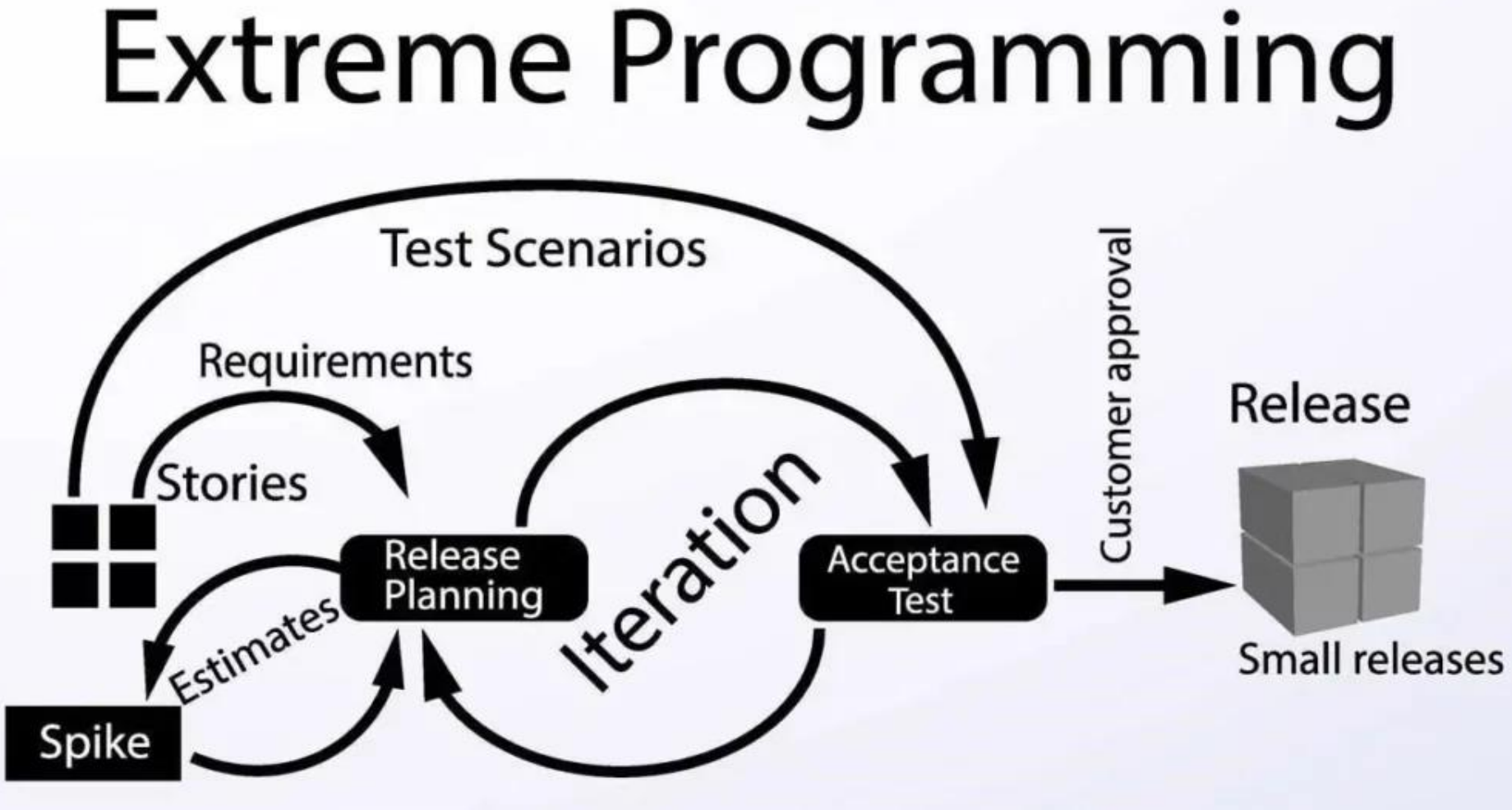
Método Ágil ➔ KANBAN

O seminário deve ser claro, conciso e objetivo, com informações precisas e exemplos relevantes para ilustrar os conceitos abordados.

Graduação Tecnológica

Engenharia de Software

Práticas de desenvolvimento de software ➔ Extreme Programming - XP



Práticas de desenvolvimento de software ➔ Extreme Programming - XP

1. Introdução ao XP

- ***O que é Extreme Programming?***

Definição e histórico.
Objetivos e benefícios.

- ***Contexto e Motivação***

Por que e quando usar XP.

2. Princípios do XP

- ***Comunicação***
- ***Simplicidade***
- ***Feedback***
- ***Coragem***
- ***Respeito***
- ***Exemplos Práticos:*** Mostrar como cada princípio é aplicado em projetos reais.

Práticas de desenvolvimento de software ➔ Extreme Programming - XP

3. Práticas do XP

- ***Planejamento de Jogo (Planning Game)***

Como planejar iterações curtas e frequentes.

Atividade Interativa: Simulação de um Planning Game

- ***Desenvolvimento Orientado a Testes (TDD)***

Importância dos testes automáticos.

Demonstração prática de TDD.

- ***Programação em Par (Pair Programming)***

Benefícios e como implementar.

Atividade Interativa: Exemplo de Pair Programming

- ***Integração Contínua***

Como garantir a integração frequente do código***.***

- ***Design Simples***

Focar em soluções simples e diretas.

Práticas de desenvolvimento de software ➔ Extreme Programming - XP

- ***Refatoração***

Melhorias contínuas no código sem alterar a funcionalidade***.***

- ***Propriedade Coletiva do Código***

Todos podem alterar qualquer parte do código.

- ***Ritmo Sustentável***

Equilibrar a carga de trabalho para evitar burnout.

- ***Metáfora***

Uso de metáforas para facilitar a compreensão do sistema.

Padronização de Código

Manter a consistência do código.

Documentação Justa

Documentação necessária, sem excessos..

Práticas de desenvolvimento de software ➔ Extreme Programming - XP

4. Aplicações e Casos de Sucesso

- ***Estudos de Caso***

Apresentar exemplos de empresas que adotaram XP com sucesso.

- ***Discussão de Resultados***

Impactos positivos observados com a implementação de XP.

5. Ferramentas de Suporte ao XP

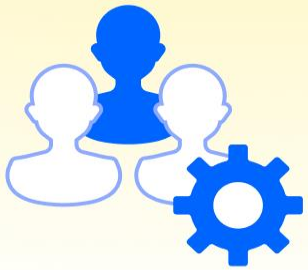
- ***Ferramentas de TDD e Integração Contínua***
- ***Ambientes de Desenvolvimento e Colaboração***

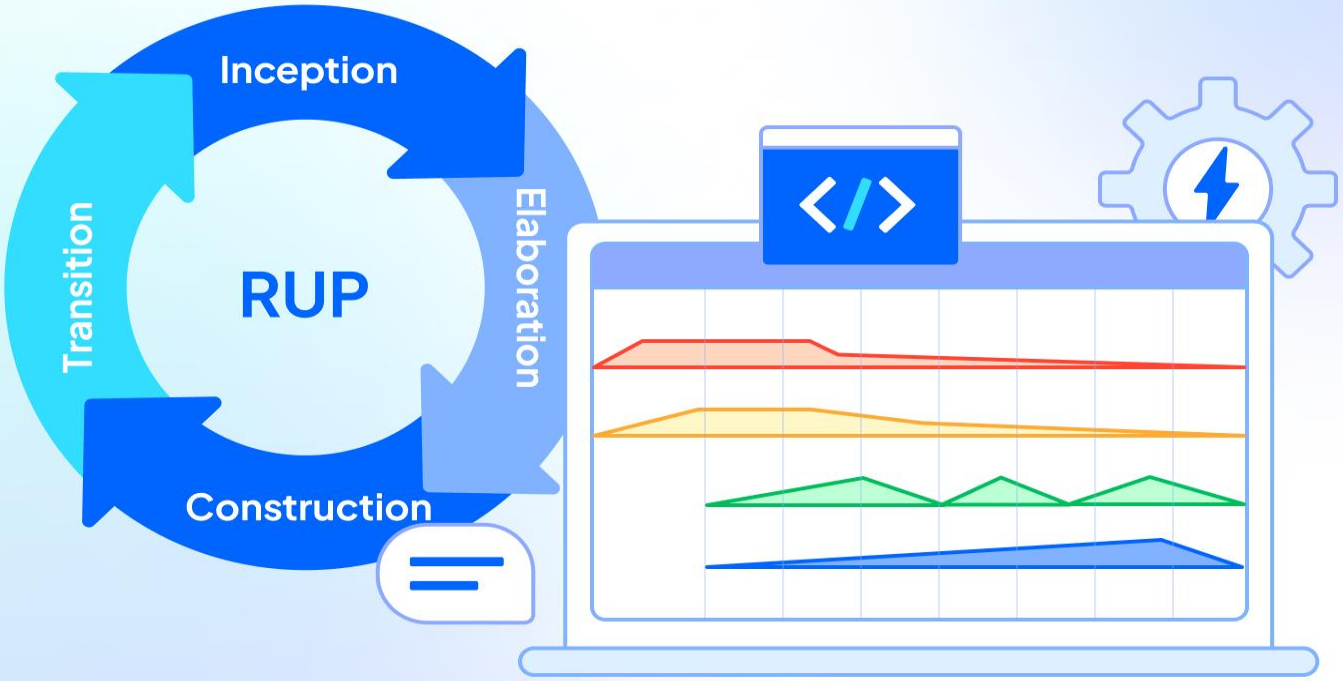
JIRA, Git, CI/CD pipelines, etc.

O seminário deve ser claro, conciso e objetivo, com informações precisas e exemplos relevantes para ilustrar os conceitos abordados.

Práticas de desenvolvimento de software ➔ Rational Unified Process - RUP

Rational Unified Process





Práticas de desenvolvimento de software ➔ Rational Unified Process - RUP

1. Introdução ao RUP:

- ***Contextualização:*** O surgimento do RUP como resposta à necessidade de um processo de desenvolvimento de software estruturado e adaptável.
- ***Definição:*** O RUP é um processo de desenvolvimento de software iterativo e incremental, focado na colaboração entre as partes interessadas.
- ***Princípios do RUP:***
  - ***Iterativo e incremental:*** O desenvolvimento ocorre em iterações curtas com entregas incrementais de valor.
  - ***Orientado a casos de uso:*** Define as funcionalidades do sistema a partir da perspectiva do usuário.
  - ***Arquitetura centrada:*** A arquitetura do sistema é definida e evolui durante o processo de desenvolvimento.

Práticas de desenvolvimento de software ➔ Rational Unified Process - RUP

- ***Gerenciamento de requisitos:*** Os requisitos são cuidadosamente coletados, analisados e gerenciados durante todo o ciclo de vida do projeto.
- ***Modelagem e visualização:*** Utiliza modelos e diagramas para comunicar o design e o funcionamento do sistema.
- ***Controle de versão:*** Gerencia as diferentes versões do código e dos documentos do projeto.
- ***Gestão de riscos:*** Identifica e gerencia os riscos que podem impactar o sucesso do projeto.

Práticas de desenvolvimento de software ➔ Rational Unified Process - RUP

2. Fases do RUP:

• ***Início:*** Define a visão do projeto, os requisitos iniciais e o escopo.

• ***Elaboração:*** Refina os requisitos, define a arquitetura do sistema e identifica os principais riscos.

• ***Construção:*** Implementa as funcionalidades do sistema, realizando testes e integrações.

• ***Transição***: Prepara o sistema para implantação, realizando testes de desempenho e treinando os usuários.

Práticas de desenvolvimento de software ➔ Rational Unified Process - RUP

3. Disciplinas do RUP:

- ***Modelagem de requisitos:*** Capturar e gerenciar os requisitos do sistema, utilizando casos de uso, diagramas de sequência e outros modelos.
- ***Projeto de análise e design:*** Definir a arquitetura do sistema, modelar as classes e os componentes, e criar diagramas de classes e de componentes.
- ***Implementação:*** Escrever o código do sistema, realizar testes unitários e integrar os componentes.
- ***Testes:*** Garantir a qualidade do software através de testes de unidade, integração, sistema e aceitação.
- ***Implantação:*** Colocar o sistema em produção, realizar a configuração e o treinamento dos usuários.
- ***Gerenciamento de configuração e mudança:*** Controlar as versões do código, dos documentos e das mudanças no sistema.
- ***Gerenciamento de projetos:*** Gerenciar o tempo, o custo, os recursos e os riscos do projeto.

Práticas de desenvolvimento de software ➔ Rational Unified Process - RUP

4. Práticas do RUP:

- ***Uso de modelos:*** Utilizar modelos para comunicar o design e o funcionamento do sistema (ex.: UML, BPMN).
- ***Gerenciamento de requisitos:*** Criar um repositório centralizado para gerenciar os requisitos e rastrear suas mudanças.
- ***Testes automatizados:*** Utilizar ferramentas para automatizar os testes de software e garantir a qualidade do código.
- ***Gerenciamento de configuração:*** Utilizar ferramentas para controlar as versões do código e dos documentos do projeto.
- ***Documentação:*** Criar documentação clara e concisa para o sistema, incluindo a documentação do código, dos requisitos, do design e da arquitetura.

Práticas de desenvolvimento de software ➔ Rational Unified Process - RUP

5. Aplicações Práticas do RUP em Desenvolvimento de Software:

- ***Desenvolvimento de um sistema de controle de acesso:*** Mostrar como o RUP pode ser utilizado para desenvolver um sistema de controle de acesso, com foco na segurança e na gestão de usuários.

6. Benefícios do RUP e desafios sobre a implementação do RUP

O seminário deve ser claro, conciso e objetivo, com informações precisas e exemplos relevantes para ilustrar os conceitos abordados.

Graduação Tecnológica

Engenharia de Software