Aula: Planejamento, Qualidade e Implantação de Software

1. Gerenciamento do Projeto de Software

Todo projeto de software precisa de um **planejamento estruturado**. Isso vai além da codificação. Envolve **cronogramas, custos, qualidade, equipe, comunicação e muito mais**.

◇ Gerenciamento do Cronograma

Criar um cronograma significa **definir as atividades, suas durações e suas dependências**. Um cronograma eficiente usa ferramentas como **Gráficos de Gantt** ou **método do caminho crítico (CPM)**. Ele garante que a entrega ocorra no tempo certo.

◇ Gerenciamento dos Custos

Controlar custos não é só calcular o valor da hora trabalhada. É prever recursos físicos, licenças de software, infraestrutura de servidores, manutenção e testes. Uma estimativa mal-feita pode levar à falência de um projeto.

◇ Gerenciamento da Qualidade

A qualidade do software não é apenas ausência de erros. Envolve **funcionalidade, usabilidade, confiabilidade e manutenção**. Gerenciar a qualidade é **criar padrões, validar entregas e ouvir o cliente**.

◇ Gerenciamento dos Recursos

Trata da **alocação inteligente da equipe**. Quem faz o quê? Quem tem mais experiência com testes? Quem lida bem com o cliente? Uma equipe mal distribuída causa retrabalho.

◇ Gerenciamento das Comunicações

Você já participou de um projeto onde ninguém sabia o que o outro estava fazendo? A comunicação precisa ser planejada: **quem se comunica com quem, como e com que frequência?** Ferramentas como Jira e Slack ajudam, mas o mais importante é a cultura de transparência.

- **Gerenciamento dos Riscos**

Todo projeto tem riscos: atrasos, falhas técnicas, mudanças no escopo. Gerenciar riscos é **antecipar cenários e criar planos B**.

- **Gerenciamento das Aquisições**

Muitas vezes, partes do sistema serão contratadas ou compradas. É necessário saber **quando terceirizar e como contratar com segurança**, incluindo cláusulas de SLA (nível de serviço).

- **Gerenciamento das Partes Interessadas (Stakeholders)**

Stakeholders são todos que afetam ou são afetados pelo projeto: **usuários, clientes, investidores, equipe técnica**. Gerenciar suas expectativas é essencial para o sucesso.

2. Identificação de Riscos

Vamos entender **de onde vêm os riscos**:

- **Tamanho do Produto**: quanto maior, mais complexo e mais chances de falha.
- **Impacto no Negócio**: sistemas críticos precisam de mais controle.
- **Características do Envolvimento**: stakeholders ausentes ou indecisos aumentam o risco.
- **Definição do Processo**: processos mal definidos geram confusão.
- **Ambiente de Desenvolvimento**: se o ambiente (infraestrutura, ferramentas) for instável, o risco aumenta.
- **Tecnologia a ser Criada**: usar algo novo e pouco testado é arriscado.
- **Quantidade de Pessoas e Experiência**: equipes grandes e inexperientes têm mais dificuldade de se coordenar.

3. Análise Qualitativa dos Riscos

Uma vez identificados, os riscos são analisados qualitativamente:

- **Probabilidade**: Qual a chance de o risco acontecer?
- **Grau de Impacto**: Se acontecer, qual o estrago?
- **Aplicação**: Usamos essas duas variáveis (probabilidade e impacto) para criar uma **matriz de risco**, ajudando a priorizar quais riscos precisam de ação imediata.

4. Fases do Desenvolvimento de Software

A Engenharia de Software se organiza em fases:

- **Engenharia de Software**

  Disciplina que aplica princípios de engenharia para projetar, desenvolver, manter e evoluir software de forma **sistematizada e controlada**.

- **Processo de Desenvolvimento de Software**

  O processo define **como** o software será desenvolvido. Pode ser tradicional (cascata) ou ágil (Scrum, XP, etc.).

- **Engenharia de Requisitos**

  É a fase onde descobrimos o **que o software precisa fazer**. Um requisito mal levantado gera retrabalho e insatisfação.

5. Qualidade de Software

Qualidade vai além de rodar sem erros. Engloba:

- **Funcionalidade**: faz o que deveria fazer?
- **Confiabilidade**: funciona bem sob pressão?
- **Usabilidade**: o usuário entende como usar?
- **Eficiência**: usa bem recursos como memória e CPU?
- **Facilidade de Manutenção**: é fácil de corrigir e melhorar?
- **Portabilidade**: roda em diferentes ambientes?

6. Teste de Software

Testar é garantir que tudo funciona como deveria. Existem vários tipos:

- **Teste de Unidade**: testa partes isoladas (funções, métodos).
- **Teste de Integração**: testa se os módulos funcionam bem juntos.
- **Teste de Validação**: verifica se os requisitos foram atendidos.
- **Teste Alpha**: feito pela equipe de desenvolvimento.
- **Teste Beta**: feito por usuários reais.
- **Aceite Formal**: quando o cliente confirma que está tudo ok.

7. Teste de Sistema

Agora testamos o sistema como um todo:

- **Teste de Recuperação**: o sistema se recupera de falhas?
- **Teste de Segurança**: protege os dados contra acessos indevidos?
- **Teste por Esforço (stress test)**: o que acontece com 1000 usuários simultâneos?
- **Teste de Desempenho**: tempo de resposta aceitável?
- **Teste de Disponibilização**: o sistema está sempre disponível?

8. Implantação

Depois de tudo validado, o sistema pode ser entregue:

- **Produção de Releases**: versões organizadas para serem publicadas.
- **Empacotamento**: montagem do instalador ou container.
- **Distribuição**: como o software chega ao usuário?
- **Instalação**: é simples? Guiada?
- **Suporte ao usuário**: tutoriais, FAQs, chat, treinamentos.

Cerimônias e Reuniões Focadas em Cronograma, Custos e Objetivos

◇ Sprint Planning (Planejamento da Sprint)

- **Objetivo:** Definir o que será feito na próxima sprint e como o trabalho será realizado.
- **Assuntos típicos:**
  - Estimativas de tempo (cronograma da sprint)
  - Capacidade da equipe
  - Priorização de tarefas conforme valor de negócio
  - Alinhamento com os objetivos do produto

◇ Release Planning (Planejamento de Release)

- **Objetivo:** Planejar entregas maiores do produto (releases), geralmente abrangendo várias sprints.
- **Assuntos típicos:**
  - Cronogramas de entrega
  - Estimativas de custo e esforço
  - Dependências entre funcionalidades
  - Objetivos estratégicos de negócio

◇ Daily Scrum (Reunião Diária ou Daily Stand-Up)

- **Objetivo:** Acompanhar o progresso diário e identificar impedimentos.
- **Assuntos típicos:**
  - Status das tarefas
  - Impactos no cronograma da sprint
  - Replanejamento leve (caso necessário)

◇ Sprint Review

- **Objetivo:** Revisar o que foi entregue e obter feedback dos stakeholders.
- **Assuntos típicos:**
  - O que foi concluído e o que não foi
  - Ajustes no backlog
  - Alinhamento com objetivos do projeto
  - Impactos em prazos futuros

◇ Sprint Retrospective

- **Objetivo:** Avaliar o processo e propor melhorias para as próximas sprints.
- **Assuntos típicos:**
  - Desvios no cronograma
  - Problemas de comunicação ou custos ocultos
  - Melhorias no uso de recursos

◇ Backlog Refinement (ou Grooming)

- **Objetivo:** Refinar e detalhar os itens do backlog para sprints futuras.
- **Assuntos típicos:**
  - Estimativas de esforço e complexidade
  - Priorização com base em valor e custo
  - Alinhamento com metas e capacidades da equipe

◇ Project Kickoff Meeting (Reunião de Início do Projeto)

- **Objetivo:** Dar início oficial ao projeto.
- **Assuntos típicos:**
  - Objetivos gerais do projeto
  - Cronograma macro e milestones

- Orçamento inicial
- Papéis e responsabilidades

◇ **Steering Committee Meeting (Reunião de Comitê de Direção)**

- **Objetivo:** Acompanhar a evolução estratégica do projeto (usada em projetos grandes).
- **Assuntos típicos:**
  - Indicadores de desempenho (tempo, custo, escopo)
  - Decisões executivas
  - Alinhamento com metas corporativas

◇ **Post-Mortem Meeting (Reunião Pós-Projeto ou Pós-Sprint)**

- **Objetivo:** Avaliar o projeto após sua conclusão.
- **Assuntos típicos:**
  - O que funcionou e o que não funcionou
  - Análise de aderência a cronograma e orçamento
  - Lições aprendidas