Resumo Engenharia de Software I - Prova 1

1. Conceitos Básicos

- **Definição**: Área da computação que aplica abordagens sistemáticas, disciplinadas e quantificáveis para desenvolver, operar, manter e evoluir software.
- **Objetivo**: Desenvolver software de forma produtiva, econômica e com qualidade, especialmente em sistemas complexos.
- **Origem**: Termo cunhado em 1968 na OTAN, em resposta à chamada “Crise do Software”.

2. Crise do Software - 1960/70

Problemas:

- Atrasos e estouros de orçamento.
- Baixa qualidade e dificuldade de manutenção.
- Dificuldade de comunicação com o cliente.

Causas:

- Desenvolvimento anárquico e improvisado.
- Falta de metodologia e documentação.
- Complexidade crescente dos sistemas.

Soluções propostas:

- Abordagem metodológica.
- Uso de ferramentas automatizáveis.
- Reuso de software.

3. Processo de Software

- Conjunto de tarefas necessárias para construir software de alta qualidade.

3.1 Ciclo de Vida Clássico:

- **Análise**: Estudo do domínio do problema e levantamento de requisitos.
- **Projeto**: Especificação da solução computacional.
- **Implementação**: Produção do código.
- **Testes**: Avaliação do código e verificação de requisitos.

- **Manutenção**: Modificações após o software estar em uso.

4. Paradigmas de Desenvolvimento

4.1 Evolução Histórica:

- **Anos 60-70**: Programação em batch, código sob encomenda, manutenção complexa.
- **Anos 80-90**: Popularização da Orientação a Objetos, padrões de projeto (Design Patterns), linguagens como Java e C++.
- **Anos 2000+**: Agile, DevOps, Clean Code, TDD, DDD, microsserviços.

Problemas Recentes:

- **Over-engineering**: Soluções complexas para problemas simples.
- **Under-engineering**: Entregas rápidas em detrimento da qualidade.
- **Vibe-coding**: Dependência excessiva de IA, sem raciocínio crítico.

5. Engenharia de Requisitos

Requisitos: Condições ou capacidades necessárias para resolver um problema ou atender a um objetivo. Servem como contrato entre cliente e equipe de desenvolvimento.

5.1 Tipos de Requisitos:

- **Funcionais**: O que o sistema deve fazer (ex.: cadastrar usuário).
- **Não Funcionais**: Restrições de como o sistema deve funcionar (ex.: performance, segurança).
- **Regras de Negócio**: Políticas ou condições da organização que influenciam o sistema.

6. Modelagem, Especificação, Validação e Verificação de Requisitos

Modelagem: Representação do domínio do problema e da solução.

Especificação: Documento claro e compreensível para clientes e desenvolvedores.

Validação: Garantir que os requisitos atendem às reais necessidades do cliente.

Verificação: Assegurar que os requisitos estão bem escritos, consistentes e completos.

7. Levantamento de Requisitos

Requisitos

- objetivos ou restrições estabelecidas por clientes e usuários do sistema que definem as diversas propriedades deste.
- não devem conter informações sobre as soluções técnicas que serão adotadas para desenvolver o sistema.

- devem ser escritos para que leitores não-técnicos consigam entender
- O documento de requisitos serve como um termo de consenso entre a equipe técnica (desenvolvedores) e o cliente. Ele estabelece o escopo do sistema.

Requisitos Funcionais (**o que**)

- todas as coisas que o sistema deve fazer

Requisitos não Funcionais (**como**)

- restrições sobre como o sistema deve realizar seus requisitos funcionais
- como o sistema se comporta

Regra de Negócio

- é uma política, condição ou restrição da organização que define COMO o negócio funciona.
- como o negócio opera

Fases do levantamento de requisitos

1. Contrato Inicial: Fechar contrato específico para o levantamento.

2. Kick-off: Reunião com equipe do cliente para alinhamento.

3. Coleta de Dados:

- Formulários com perguntas estratégicas.
- Documentos com necessidades por área.

4. Análise e Validação:

- Organizar requisitos por módulos.
- Validação com o dono do negócio.

5.Documento Final:

- Anexado ao contrato principal.
- Serve como escopo legal.

6. Orçamento e Negociação:

- Preço justo, sem desespero.
- Bônus em vez de desconto.

7. Desenvolvimento: Entregas parciais com aprovação do cliente.

8. Metodologias Ágeis

8.1 Manifesto Ágil (2001):

- Indivíduos e interações > processos e ferramentas.

- Software funcionando > documentação extensiva.
- Colaboração com o cliente > negociação de contratos.
- Responder a mudanças > seguir um plano.

8.2 Práticas Comuns:

- Iterações curtas (sprints).
- Entregas contínuas.
- Feedback constante.

8.3 Problemas Atuais:

- Distorção do ágil para justificar prazos irreais.
- Superficialidade técnica.

9. Modelos de Processo de Software

9.1 Modelo em Cascata (Sequencial Linear)

Características:

- Fases sequenciais e lineares.
- Uma fase só começa quando a anterior termina.
- Fases típicas: Requisitos → Projeto → Implementação → Testes → Manutenção.

Vantagens:

- Simples e fácil de gerenciar.
- Documentação robusta.
- Bom para requisitos bem compreendidos e estáveis.

Desvantagens:

- Pouca flexibilidade para mudanças.
- Cliente só vê o produto no final.
- Problemas são detectados tardiamente.

9.2 Modelo Incremental

Características:

- O software é desenvolvido e entregue em partes (incrementos).

- Cada incremento adiciona novas funcionalidades.
- Combina elementos do cascata com iterações.

Vantagens:

- Entrega parcial de valor ao cliente.
- Melhor gerenciamento de riscos.
- Flexibilidade para incorporar feedback.

Desvantagens:

- Requer planejamento cuidadoso dos incrementos.
- Arquitetura deve ser bem definida desde o início.

9.3 Modelo RAD (Desenvolvimento Rápido de Aplicações)

Características:

- Foco em desenvolvimento rápido e entrega em curto prazo.
- Uso intensivo de prototipagem e ferramentas CASE.
- Equipes pequenas e especializadas.

Vantagens:

- Redução significativa do tempo de desenvolvimento.
- Maior envolvimento do cliente.

Desvantagens:

- Requer equipe experiente.
- Só funciona para projetos modulares e escopo bem delimitado.

9.4 Modelo de Prototipagem

Características:

- Construção de protótipos para explorar requisitos.
- Cliente interage com o protótipo e dá feedback.
- Iterativo até chegar a um sistema aceitável.

Vantagens:

- Reduz ambiguidades nos requisitos.

- Cliente participa ativamente.

Desvantagens:

- Pode levar a escopo aberto e sem fim.
- Cliente pode achar que o protótipo é o produto final.

9.5 Modelo Espiral

Características:

- Combinatório: cascata + prototipagem + análise de riscos.
- Dividido em ciclos (espirais), cada um com quatro fases:
    - Determinação de objetivos.
    - Análise e redução de riscos.
    - Desenvolvimento e validação.
    - Planejamento do próximo ciclo.

Vantagens:

- Gerenciamento proativo de riscos.
- Flexível a mudanças.

Desvantagens:

- Complexo e caro.
- Requer experiência em gerenciamento de riscos.

9.6 Modelo de Desenvolvimento Concorrente

Características:

- Atividades ocorrem simultaneamente.
- Estados das atividades: Espera, Em execução, Revisão, Atrasada, Concluída.
- Comunicação constante entre as equipes.

Vantagens:

- Reflexão do mundo real (paralelismo).
- Entrega mais rápida.

Desvantagens:

- Complexo de gerenciar.
- Requer boa comunicação e sincronização.

9.7 Desenvolvimento Baseado em Componentes

Características:

- Foco na reutilização de componentes de software.
- Desenvolvimento a partir de componentes existentes.
- Atividades: Pesquisa, adaptação e integração de componentes.

Vantagens:

- Redução de tempo e custo.
- Maior confiabilidade (componentes já testados).

Desvantagens:

- Dependência de componentes externos.
- Pode levar a compromissos na arquitetura.

9.8 Modelo de Métodos Formais

Características:

- Uso de técnicas matemáticas para especificação e verificação.
- Especificação formal, verificação formal e transformação verificada.
- Foco na correção e ausência de ambiguidades.

Vantagens:

- Alta confiabilidade e precisão.
- Ideal para sistemas críticos.

Desvantagens:

- Curva de aprendizado íngreme.
- Consome tempo e recursos.
- Dificuldade de comunicação com clientes não técnicos.

10. Levantamento de Requisitos

Ciclo de Vida do Software

Domínio do Problema: a parte do mundo real que é relevante ao desenvolvimento de um sistema

1. Análise: estudar e descrever o domínio do problema

2. Projeto: estudar e descrever o domínio da solução

3. Implementação: produção do código

4. Testes: avaliação do código produzido

5. Manutenção: modificação do software original (não se resume a alterações de código)

11. Modelos de Tabelas

Para Requisitos Funcionais

RF1	**nome**
DESCRIÇÃO	...
PRIORIDADE	...

Para Requisitos Não funcionais:

RNF1	**nome**
DESCRICAO	...
CATEGORIA	...
OBRIGATORIEDADE	...

Para Regras de Negócio

Nome	**...**
DESCRIÇÃO	...
FONTE	...
HISTÓRICO	Data de identificação: ...