**Resumo ESI – P1**

**Engenharia de Software** : é uma abordagem sistemática baseada em processos, procedimentos, ferramentas e técnicas para produzir sistemas computacionais que possuem requisitos de qualidade observáveis.

Para análise do problema, o mesmo é subdividido em problemas menores (subproblemas). Já para a síntese de soluções, as soluções parciais são combinadas para formar uma solução única total.

**Processos**: seqüência de atividades a serem realizadas para construir um sistema de software.

**Método ou técnica:** procedimento formal para a produção de um resultado.

**Ferramenta:**  instrumento ou sistema automatizado para realizar alguma coisa.

**Procedimento:** receita de combinação de ferramentas e técnicas.

**Paradigma:** estilo de fazer algo.

A Ciência da Computação busca solucionar problemas genéricos. Já a Engenharia de Software escolhe o melhor algoritmo entre os existentes para resolver o problema do cliente. Utiliza técnicas para reduzir o problema do cliente em problemas menores e mais gerenciáveis, utiliza técnicas para garantir a qualidade do produto.

**Perspectivas de Qualidade**

Visão transcendental: algo que podemos reconhecer mas não definir.

Visão do usuário: adequação ao propósito pretendido.

Visão do Fabricante : conformidade com especificação.

Visão do Produto: relação com as características inerentes ao produto (organização, eficiência)

Visão do Mercado : dependência de quanto os consumidores estão dispostos a pagar.

Visão Interna : facilidade de manutenção, com códigos claros e com facilidade de testar.

A comunicação entre clientes, usuários e desenvolvedores é a chave do desenvolvimento de um bom software. Cliente: quem paga o sistema; Usuário : quem vai usar o sistema; Desenvolvedor: quem faz o software.

Um sistema é a união de : hardware, software, usuário, conexões com outros hardwares e softwares, onde é importante estabelecer as fronteiras.

**Processo** : envolve uma série de etapas que envolvem atividades, restrições e recursos para alcançar a saída desejada. Inclui produtos intermediários e finais além de subprocessos com hierarquia , organizados de algum modo. Modela – se um processo com o objetivo de formar um entendimento comum, encontrar inconsistências, redundâncias e omissões, avaliar atividades propostas mais adequadas aos objetivos e fazer um processo geral para uma situação particular na qual ele será utilizado.

**Processo de Desenvolvimento de Software** : envolve todas as atividades necessárias para definir, desenvolver, testar e manter um produto de software. Participam : cliente , analista, projetista, gerente de projeto, programadores, arquitetos de software, avaliadores de qualidade e outros.

Atividades fundamentais que são comuns a todos os processos de software:

1. **Especificação do Software** : clientes e engenheiros definem o software a ser produzido e as restrições para a sua operação.
2. **Desenvolvimento de Software** : o software é projetado e programado.
3. **Validação de Software** : o software é verificado para garantir que é o que o cliente deseja.
4. **Evolução de Software** : o software é modificado para se adaptar às mudanças dos requisitos do cliente e do mercado.

**Objetivos de um processo de desenvolvimento** : definir as atividades que serão executadas ao longo do desenvolvimento, definir como, quando e por quem essas atividades serão executadas, prover pontos de controle para verificação do andamento do desenvolvimento e obter a padronização da forma de desenvolver software em uma organização.

**Requisitos Funcionais** : as funções que o sistema deve executar.

**Requisitos Não Funcionais** : as restrições sob as quais o sistema deve operar.

**Requisitos de Domínio** : as características e restrições do domínio da aplicação do sistema. Podem ser requisitos funcionais ou não funcionais.

A fase de projeto enfatiza a proposta de uma solução que atenda os requisitos da análise. Então, se a análise é uma investigação para tentar descobrir o que o cliente quer, o projeto consiste em propor uma solução com base no conhecimento adquirido na análise.

Inspeções e revisões devem ser feitas a cada estágio do processo de software, desde a definição de requisitos de usuário até a implantação do software.A maior parte dos custos de validação ocorrem quando o software está sendo testado, após a implementação.

**Verificação e Validação (Testes)**

Teste de componente(ou unidade) : os componentes individuais são testados para garantir que operem corretamente. Cada componente é testado independentemente, sem os outros componentes do sistema.

Teste de Integração : os componentes são integrados para compor o sistema. Busca erros que resultam das interações não previstas entre os componentes e problemas de interface de componentes.

Teste de Sistema: o sistema é testado para garantir que o projeto foi corretamente construído, fornecendo as saídas corretas de acordo com a especificação.

Teste de Aceitação : é o estágio final do processo de teste, antes que o sistema seja aceito para o uso operacional. O sistema é testado com os dados fornecidos pelo cliente do sistema, em vez de dados simulados de teste.

**Modelos de Processo de Software**

* **Modelo em Cascata:**

Análise dos Requisitos 🡪 Projeto do Sistema 🡪 Projeto do Programa 🡪 Codificação 🡪 Teste de Unidades e Integração 🡪 Teste do Sistema 🡪 Teste de Aceitação 🡪 Operação e Manutenção

**Problemas :** Os processos raramente seguem um fluxo seqüencial, é difícil de definir todas as restrições a priori, a primeira versão em estágio tardio.

Apesar dos problemas , foi largamente utilizado, é melhor que não ter uma sistemática de desenvolvimento.

Pode ser usado com prototipação : um produto é parcialmente desenvolvido, possibilita aos clientes e desenvolvedores examinarem certos aspectos do sistema e decidir se eles são ou não apropriados ou adequados para o produto acabado. Eles ajudam ao projetista a avaliar estratégias alternativas do projeto e decidir qual é a melhor para cada projeto.Freqüentemente a interface do usuário é construída e testada como um protótipo. Os principais problemas são tratados e corrigidos bem antes de serem oficialmente validados no teste de sistema.

**Problemas da Prototipação :** usuários enxergam o protótipo como o produto final.As ferramentas e estruturas utilizadas podem se tornar uma versão final e a manutenção do protótipo tende a ser problemática.

Modelo em V : é uma variação do modelo em cascata que mostra como as atividades de teste estão relacionadas com a análise e projeto. Neste modelo os teste de modelo e integridade garantem a qualidade dos programas. Os testes de aceitação são conduzidos mais pelos clientes do que pelos desenvolvedores, valida os requisitos.

* **Desenvolvimento Iterativo e Incremental :**

Iterativo : o sistema de software é desenvolvido em vários passos similares.

Incremental : em cada passo o sistema é estendido com mais funcionalidades.

Divide o desenvolvimento de um produto de software em ciclos. Em cada ciclo de desenvolvimento podem ser identificadas as fases de análise, projeto, implementação, testes e implantação. Cada ciclo considera um subconjunto de requisitos. Contrasta com a abordagem clássica, onde as fases são realizadas de uma única vez.

Este modelo incentiva a participação do usuário, os riscos do desenvolvimento podem ser bem mais gerenciados.É mais difícil de gerenciar, possibilidade de ciclos em paralelo.

**Risco de Projeto :**  é a possibilidade de ocorrência de algum evento que cause prejuízo ao processo de desenvolvimento, juntamente com as conseqüências desse prejuízo. Influências : custos, cronograma, qualidade, satisfação.

* Modelo Formal : um modelo matemático do sistema é formalmente transformado em uma implementação.

**Problemas :**  apenas especialistas são capazes de empregar a técnica, requer treinamento em linguagens formais.

**Aplicação :** sistemas críticos, especialmente aqueles onde a segurança deve ser assegurada e garantida, antes do sistema ser posto em operação.

* **Engenharia de Software baseada em componentes (CBSE)** : supõe que partes do sistema já existem. O processo de desenvolvimento concentra-se mais na integração dessas partes do que no seu desenvolvimento a partir do início.

Abordagem baseada em reuso para definição, implementação e composição de componentes independentes não acoplados nos sistemas. Explícita separação entre a especificação e a implementação das funcionalidades. Os detalhes de implementação não são visíveis aos usuários das funcionalidades. Há flexibilidade e facilidade de manutenção – incorporação de mudanças ou inclusão de novas funcionalidades com mínimo impacto nas outras partes do sistema.

Apresenta composição uniforme : a composição é adquirida através de conectores que reconhecem as interfaces dos componentes e fazem a adaptação de interfaces. Apresenta interoperabilidade e reusabilidade.

**Componente :** é uma unidade de software cuja funcionalidade e dependências são completamente definidas por um conjunto de interfaces públicas.

**Interfaces Providas :** serviços que o componente oferece.

**Interfaces Requeridas :** serviços que o componente necessita para prover os seus serviços.(interfaces providas)

**Casos de Uso**

São uma ferramenta de levantamento de requisitos funcionais de um sistema de software a ser desenvolvido.

O modelo conceitual e os contratos estão diretamente ligados ao projeto de camada de domínio.

**Estória de Usuário** : é um exemplo de utilização do caso de uso, serve como um aquecimento para a descrição propriamente dita do caso de uso

**Ator** : qualquer coisa que possui um comportamento. Pode ser uma pessoa, uma organização, programa ou sistema computacional, hardware.

**Documento de Especificação de Requisitos de Software**

Os pontos básicos que devem ser definidos no documento são: funcionalidade, interfaces externas, restrições , critérios de qualidade e de desempenho.O documento deve definir quais funções são executadas sobre que dados para produzir quais resultados em qual local e para quem.

Toda equipe deve participar da elaboração do documento, que deve definir corretamente todos os requisitos de software, mas sem descrever detalhes do projeto ou de implementação. O documento descreve as necessidades do produto de software mas não o processo de produção.

Uma especificação de requisitos é **completa** se contém todos os requisitos importantes, definindo todas as definições de respostas de software para todas as classes imagináveis de entrada de dados em todas as situações imagináveis.

O documento deve especificar todas as necessidades que poderiam ser atendidas pelo software, classificando – as de acordo com a importância em: **Essenciais**  (atendimento obrigatório), **Desejáveis**  (atendimento fortemente recomendado) e **Opcionais** (seria bom atender).

Um requisito deve ser verificável, ou seja, deve haver uma maneira de garantir que o requisito é atendido pelo software.

**O documento deve conter :**

* Uma introdução falando do propósito do documento, estabelecendo o objetivo e o público alvo. Além disso, deverá ter um escopo, que define os produtos especificados no documento tais como : nome, objetivo, limites de atuação e benefícios esperados, além de uma visão geral do documento.
* Após, uma parte de suporte onde são listados todos os documentos citados no documento de especificação de requisitos e as explicações de todas as siglas e abreviações encontradas.
* Uma descrição geral do software contendo o contexto em que o produto será executado: interconexões, interfaces externas, operações, sistemas de comunicação, restrições de memória (primária e secundária) e requisitos de plataforma.Além disso, conterá especificações das principais funções do software, o perfil de usuário esperado, restrições , obrigações ou limitações às opções de desenvolvimento e os requisitos futuros que não foram contemplados na versão atual do software.
* Requisitos específicos : definir detalhes de intercâmbio de dados com entidades externas. Definir usuários, hardware, software , sistemas de comunicação.
* Definir para cada função entradas válidas e inválidas, operações realizadas, resultados ou saídas aceitáveis e tratamento de exceções.
* Requisitos de desempenho estáticos (quantidade de determinada rede, conexões simultâneas, volume de dados armazenados ) e dinâmicos (transações por segundo, tempos de respostas, memória).
* Requisitos de bases de dados lógicas, restrições de projeto e atributos do sistema de software (fatores de qualidade, portabilidade, segurança, manutenibilidade).
* Apêndice : contendo comentários adicionais e documentos importantes (contendo leis, restrições, regulações)

**Engenharia de Requisitos** : processo de descobrir, analisar, documentar e verificar serviços oferecidos e restrições do sistema.

**Tipos de Requisitos**

* **Requisitos de Usuário** : declarações, em linguagem natural com diagramas, de quais serviços são esperados do sistema e as restrições sob as quais ele deve operar. Público alvo: clientes. Devem descrever os requisitos funcionais e não funcionais de modo que os usuários que não possuem conhecimento técnico detalhado possam entender.
* **Requisitos de Sistema** : Definem, detalhadamente, as funções, os serviços e as restrições operacionais do sistema. Deve de ser preciso, deve definir exatamente o que será implementado. Devem ser uma especificação completa e consistente de todo o sistema.
* **Requisitos Funcionais** : são declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações.Em alguns casos podem também especificar o que o sistema não deve fazer. A declaração de requisitos funcionais de um sistema deve de ser completa (todos os serviços exigidos pelo usuário devem ser definidos) e consistente (os requisitos não devem ter definições contraditórias).
* **Requisitos Não Funcionais** : são restrições sobre os serviços ou as funções oferecidas pelo sistema.São diretamente relacionados às funções específicas fornecidas pelo sistema. São geralmente mais importantes que os funcionais.
* **Requisitos de Domínio** : são provenientes do domínio da aplicação do sistema e refletem as características e as restrições desse domínio. Podem ser requisitos funcionais ou não funcionais.
* **Requisitos do Produto** : especificam o comportamento do produto.(confiabilidade, usabilidade)
* **Requisitos Organizacionais** : são derivados de políticas e procedimentos da organização do cliente e do desenvolvedor.(implementação, entrega)
* **Requisitos Externos**: é amplo, envolve todos os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento.
* **Requisitos de Projeto** : descritos em linguagem que subsidie o desenvolvimento do projeto. Pode ser utilizado como parte de um contrato entre cliente e desenvolvedores. Podem ser descritos com notação adicional como casos de uso, diagramas de fluxos e especificações formais.

Notações dinâmicas de requisitos: tabelas de decisão, máquinas de estados finitos, diagramas de estados, diagramas de seqüências.

**Teste de Software** : processo de executar um programa com o objetivo de revelar a presença de defeitos; ou, falhando nesse objetivo, aumentar a confiança sobre o programa.

**Defeito** : deficiência mecânica ou algorítmica que se ativada pode levar a uma falha.

**Erro**  : ocorre quando o valor esperado e o valor obtido não são os mesmos.

**Falha:**  produção de uma saída incorreta sem relação à especificação.

**Casos de Teste** : especificação de uma entrada para o programa e a correspondente saída esperada.Seu conjunto deve ser finito e seu custo de aplicação deve ser razoável.

**Teste de Unidade**: identificar erros de lógica e de implementação em cada módulo do software, separadamente.

**Teste de Integração**: identificar erros associados às interfaces entre módulos do software

**Teste de Validação** : verificar se os requisitos estabelecidos na análise de requisitos são atendidos pelo software. Garantir que o software atende a todas as exigências funcionais, comportamentais e de desempenho.

**Teste de Sistema:** verificar se os elementos do sistema combinam –se adequadamente e se a função/desempenho global do sistema é conseguida.