Software

São os programas de computador e a documentação associada. Produtos de software podem ser desenvolvidos para um cliente específico ou para o mercado.

Engenharia de Software

Engenharia de software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software.

Qual é a diferença entre Engenharia de Software e Ciência da Computação?

A ciência da computação se ocupa da teoria e dos fundamentos; a engenharia de software trata da praticidade de desenvolver e fornecer software útil.

Qual é a diferença entre engenharia de software e engenharia de sistemas?

A engenharia de sistemas se ocupa de todos os aspectos relacionados ao desenvolvimento de sistemas com base em computadores, incluindo hardware, software e engenharia de processos. A engenharia de software é parte desse processo.

O que é um processo de software?

É um conjunto de atividades, cuja meta é o desenvolvimento ou a evolução do software.

1. ***Especificação do software*** - A funcionalidade do software e as restrições em sua operação devem ser definidas.
2. ***Desenvolvimento do software*** - O software deve ser produzido de modo que atenda a suas especificações.
3. ***Validação do software*** - O software tem de ser validado para garantir que ele faz o que o cliente deseja.
4. ***Evolução do software*** - O software deve evoluir para atender às necessidades mutáveis do cliente.

O que é um modelo de processo de software?

É uma representação simplificada de um processo de software, apresentada a partir de uma perspectiva específica.

Quais são os custos da engenharia de software?

Aproximadamente 60 por cento dos custos são relacionados ao desenvolvimento e 40 por cento são custos referentes aos testes. Para o software personalizado, os custos de evolução frequentemente excedem os custos de desenvolvimento.

O que são métodos de engenharia de software?

São abordagens estruturadas para o desenvolvimento de software, que incluem modelos de sistemas, notações, regras, recomendações de projetos e diretrizes de processos.

O que é CASE (computer-aided software engineering — engenharia de software com o auxílio de computador)?

São sistemas de software destinados a proporcionar apoio automatizado às atividades de processo de software. Os sistemas CASE são freqüentemente utilizados para proporcionar apoio aos métodos.

Quais são os atributos de um bom software?

O software deve proporcionar ao usuário a funcionalidade e o desempenho requeridos e deve ser passível de manutenção, confiável e de fácil uso.

Quais são os principais desafios enfrentados pela engenharia de software?

Lidar com sistemas legados, atender à crescente diversidade e atender às exigências quanto a prazos de entrega reduzidos.

Responsabilidade profissional e ética do ES

1- ***Confidencialidade*** - Os engenheiros devem, normalmente, respeitar a confidencialidade de seus empregadores ou clientes, quer tenham ou não assinado um acordo formal de confidencialidade.
2- ***Competência*** - Os engenheiros não devem enganar quanto ao seu nível de competência. Não devem conscientemente aceitar serviços que estejam fora do seu limite de competência.
3- ***Direitos de propriedade intelectual*** - Os engenheiros devem estar cientes das leis locais que regulam o uso da propriedade intelectual, como patentes e direitos autorais. Eles devem ser cuidadosos, a fim de assegurar que a propriedade intelectual de empregadores e clientes seja protegida.
4- ***Má utilização de computadores*** - Os engenheiros de software não devem empregar suas habilidades técnicas para o mau uso dos computadores de outras pessoas. O mau uso de computadores abrange desde casos relativamente triviais (por exemplo, jogar no

computador do empregador) até casos extremamente sérios (como a disseminação de vírus).

Processo da Engenharia de Requisitos

Designa todas as atividades envolvidas em descobrir, obter, analisar, especificar, documentar, verificar, gerir, manter os requisitos dum sistema.

Fases Principais do Processo de Engenharia de Requisitos:

1- ***Estudo de Viabilidade*** - É feita uma estimativa para verificar se as necessidades dos usuários que foram identificadas podem ser satisfeitas com a utilização das atuais tecnologias de software e hardware.
2- ***Levantamento e análise de requisitos*** - Este é o processo de obter os requisitos do sistema pela observação de sistemas existentes, pela conversa com usuários e compradores em potencial, pela análise de tarefas e assim por diante.
3- ***Especificação de requisitos*** - É a atividade de traduzir as informações coletadas durante a atividade de análise em um documento que defina um conjunto de requisitos.
4- ***Validação de requisitos*** - Essa atividade verifica os requisitos quanto a sua pertinência, consistência e integralidade.

Técnicas de Descoberta de Requisitos

- ✓ Analise de documentação;
- ✓ Analise de sistemas existentes;
- ✓ Entrevistas;
- ✓ Encontros;
- ✓ Cenários;
- ✓ Protótipos;
- ✓ Reutilização de Requisitos.

Arquitectura de Sw

Projeto de arquitetura está preocupado com a compreensão de como o sistema deve ser organizado e com a estrutura geral desse sistema.

Vantagens da arquitetura:

- ✓ Pode ser usada como foco de discussão com stakeholders.

Niveis de Abstração:

1- ***Arquitetura em pequena escala*** - está preocupada com a arquitetura de programas Individuais. Nesse nível, estamos preocupados com a maneira como um programa individual é decomposto em componentes.
2- **Arquitetura em grande escala** - preocupa-se com a arquitetura de sistemas corporativos complexos que incluem outros sistemas, programas e componentes de programas.

Decisões de projeto de arquitetura:

1- Existe uma arquitetura genérica de aplicação que pode actuar como um modelo para o sistema que está sendo projetado?
2- Como o sistema será distribuído? Por meio de um número de núcleos ou processadores?
3- Que padrões ou estilos de arquitetura podem ser usados?
4- Qual será a abordagem fundamental para se estruturar o sistema?
5- Como os componentes estruturais do sistema serão decompostos em subcomponentes?
6- Que estratégia será usada para controlar o funcionamento dos componentes do sistema?
7- Qual a melhor organização de arquitetura para satisfazer os requisitos não funcionais do sistema?
8- Como o projeto de arquitetura será avaliado?
9- Como a arquitetura do sistema deve ser documentada?

Visões da arquitetura

É impossível representar todas as informações relevantes sobre a arquitetura de um sistema em um único modelo de arquitetura, pois cada modelo mostra apenas uma visão ou perspectiva do sistema:

1- **A visão lógica -** que mostra as abstrações fundamentais do sistema como objetos ou classes de objetos. Nesta Visão, deve ser possível relacionar os requisitos de sistema com as entidades.
2- **A visão de processo -** que mostra os processos e componentes que compõem o sistema em tempo de execução. Esta visão é útil para analisar as características não funcionais do sistema, como desempenho e disponibilidade.
3- **A visão de desenvolvimento -** que mostra como o software é decomposto para o desenvolvimento, ou seja, apresenta a distribuição do software em componentes que são implementados por um único desenvolvedor ou por uma equipe de desenvolvimento. Essa visão é útil para gestores de software e programadores.
4- **Uma visão física** - que mostra o hardware do sistema e como os componentes de software são distribuídos entre os processadores. Essa visão é útil para os engenheiros de sistemas que estão planeando uma implantação do sistema.

Padrões da arquitetura

Um padrão de arquitetura é uma descrição estilizada das boas práticas de projecto, que tem sido experimentadas e testadas em diferentes ambientes.

▪ MVC

▪ Modelo em Camadas

▪ Arquitectura de Repositório

▪ Arquitectuta Cliente-Servidor

MVC

O sistema é estruturado em três camadas que interagem entre si, a camada modelo que faz a gestão das operações associadas aos dados, a camada visão que gere e define como os dados serão apesentados ao utilizador e por fim a componente Controlador que faz a gestão das interações do utilizador e associa-as a visão e ao modelo.

Modelo em Camadas

Organiza o sistema em um conjunto de camadas (ou máquinas abstratas) cada uma das quais fornecem um conjunto de serviços.

Arquitectura de Repositório

Todos os dados em um sistema são geridos em um repositório central, acessível a todos os componentes do sistema. Os components não interagem diretamente, apenas por meio do repositório.

Arquitectuta Cliente-Servidor

A funcionalidade do sistema está organizada em serviços, cada serviço é prestado por um servidor. Os clientes são os utilizadores desses serviços e acedem aos servidores para fazer uso dos serviços.

Arquitecturas de Aplicação

Uma arquitetura genérica de aplicação é uma arquitetura para um tipo de sistema de software que pode ser configurada e adaptada para criar um sistema que atenda aos requisitos específicos.

Pode ser usada como:

- ✓ Como ponto de partida para o projeto de arquitetura.
- ✓ Como uma forma de organizar o trabalho da equipe de desenvolvimento.
- ✓ Como uma forma de avaliar componentes para reutilização

Sistemas de Processamento de Transacções

São desenhados para processar pedidos de utilizadores para acesso a informação de uma BD, ou pedidos de actualização de uma BD.

Sistemas de Informação

Permitem acesso controlado a grandes bases de dados, são sistemas baseados em transações pois geralmente a interação com esses sistemas envolve transações de base de dados.

Sistemas de Processamento de Linguagens

Traduz linguagem natural ou artificial language para outra representação dessa linguagem e, para linguagens de programação, o código resultante pode ser executado.

Reusabilidade

É uma abordagem de desenvolvimento que tenta maximizar o reúso de softwares existentes.

3 factores para considerar um componente reusável

- ✓ Diferentes contextos
- ✓ Equipas Diferentes
- ✓ Perspectivas Diferentes
- ✓ Reusar software consiste em:
- ✓ Reusar aplicações inteiras;
- ✓ Reusar componentes;
- ✓ Reusar objectos , funções ou bibliotecas.

Vantagens

- ✓ Confiança aumentada
- ✓ Risco de processo reduzido
- ✓ Uso eficaz de especialistas
- ✓ Conformidade com padrões
- ✓ Desenvolvimento acelerado

Desvantagens

- ✓ Confiança aumentada
- ✓ Risco de processo reduzido
- ✓ Uso eficaz de especialistas
- ✓ Conformidade com padrões

- ✓ Desenvolvimento acelerado

Framework - um conjunto integrado de artefatos de software (como classes,objetos e componentes) que calaboram para fornecer uma arquitectura reusável para uma família de aplicações relacionadas.

Recursos apresentados por alguns frameworks para aplicações

Web:

- ✓ **Proteção** – classes para ajudar a implementar a autenticação de utilizador (login) e controle de acesso, para garantir que os utilizadores só possam acessar a funcionalidade permitida no sistema.
- ✓ **Páginas Web dinâmicas -** classes para ajudar na definição de templates de páginas Web, preenchê-los dinamicamente, com dados específicos da base de dados do sistema.
- ✓ **Suporte de base de dados** - Geralmente, os frameworks não incluem uma base de dados, mas assumem uma base de dados separada, como MySQL por exemplo.
- ✓ **Gestão de sessão** – Classes para criar e gerir sessões(um número de interações com o sistema ou utilizador)
- ✓ **Interação de utilizador** - Atualmente, a maioria dos frameworks Web fornece suporte a AJAX que permite que sejam criadas mais páginas Web interativas

<u>Qualidade</u>

É a totalidade de características de uma entidade que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas.

- ✓ **Explícitas (externas)** - condições em que produto deve ser usado, objetivos, funções, desempenho esperado (depende de especificações de requisitos).
- ✓ **Implícitas (internas)** - Não estão especificadas nos requisitos, mas são características obvias e fundamentais.

<u>Visões da qualidade</u>

- ✓ **Utilizador:**

1- Interessado na qualidade de uso e desempenho
2- Interessado nas medidas externas:

3- Software é confiável?

- ✓ **Desenvolvedor**

4- Interessado na Qualidade de produtos intermediários (documentos, modelos e diagramas.

5- Coerente com expectativas dos utilizadores (requisitos e aceitação)

- ✓ **Gestor de Projecto**

6- Interessado na relação Qualidade x Prazo x Custos.

7- Interessado na Qualidade do processo.

Características da Qualidade:

- Funcionalidade - Ser adequado aos requisitos.

- Confiabilidade - Manter o nível de desempenho.

- Usabilidade - Ser de fácil uso, com menor esforço.

- Manutenibilidade - ser fácil de manter

- Portabilidade - Ser transferivel de ambiente

- Eficiência - Uso otimizado de recursos

- Eficácia - Permitir que utilizador atinja sua meta com acurácia e completude

- Produtividade - Permitir que os utilizadores empreguem a quantidade apropriada de recursos em relação a eficácia obtida

- Segurança - Apresentar níveis aceitáveis de riscos

- Satisfação - Satisfazer aos utilizadores.

Evolução de software - é o acto de gerir espectativas e requisitos mutáveis dos clientes e do negócio.

É importante pq responde pelas necessidades empresariais em constante mudança, por relatos de defeitos de software ou por alterações de outros sistemas em um ambiente de software.

Engenharia de software é um processo em espiral com requisitos, projeto, implementação e testes que dura toda avida útil do sistema.

EVOLUCAO DE SOFTWARE SEGUNDO RAJILITCH E BENNET

Propuseram uma visão alternativa do ciclo de vida de evolução do software. Distinguem as fases de evolução, em **serviço e interupção gradual**:

A **evolução** é a fase em que mudanças significativas na arquitetura e funcionalidade do software podem ser feitas.

1- **Em serviço**, as únicas mudanças feitas são relativamente pequenas,essenciais. Durante essa fase, a empresa está considerando como o software pode ser substituido.
2- Na **interrupção gradual**, o software ainda pode ser usado, mas nao sao Implementadas novas mudanças. Os usuários precisam contornar qualquer problema que descubram.

PROCESSO DE EVOLUCAO

A evolução dos processos de software pode variar dependendo do tipo de software que esteja sendo mantido, dos processos de desenvolvimento usados e das habilidades das pessoas envolvida.

Propostas de mudança podem vir de requisitos já existentes que não tenham sido implementados no release de sistema, solicitações de novos requisitos, relatórios de bugs do sistema apontados pelos stackholders e novas ideias para melhoria do software vindas da equipe de desenvolvimento.

LEIS DE LEHMAN

- ✓ A **primeira lei** afirma que a manutenção de sistema é um processo inevitável. Como o ambiente do sistema muda, novos requisitos surgem, e o sistema deve ser modificado. Quando o sistema modificado é reintroduzido no ambiente, este promove mais mudanças no ambiente, de modo que o processo de evolução recomeça.

- ✓ A **segunda lei** afirma que, quando um sistema é alterado, sua estrutura se degrada. A única maneira de evitar que isso aconteça é investir em manutenção preventiva. Você gasta tempo melhorando a estrutura do software sem aperfeiçoar sua funcionalidade. Obviamente, isso implica custos adicionais, além da implementação das mudanças de sistema exigidas.
- ✓ A **terceira lei** é, talvez, a mais interessante e a mais controversa das leis de Lehman. Ela sugere que sistemas de grande porte têm uma dinâmica própria, estabelecida em um estágio inicial do processo de desenvolvimento. Isso determina a tendência geral do processo de manutenção de sistema e limita o número de possíveis alterações no mesmo.

MANUTENÇÃO

É o processo geral de mudança em um sistema depois que ele é liberado para uso.

As alterações feitas no software podem ser simples mudanças para correção de erros de codificação, até mudanças mais extensas para correção de erros de projeto, ou melhorias significativas para corrigir erros de especificação ou acomodar novos requisitos. ~

TIPOS DE MANUTENÇÃO

CORRECTIVA - Correção de erros de:

 codificação (sao baratos)

 projecto (sao caros, implicam reescrever varios componentes do programa)

 requisitos(sao mais caros, implicam reescrever o projecto).

PREVENTIVA

EVOLUTIVA - é necessário quando os requisitos de sistema mudam em resposta às mudanças organizacionais ou de negócios.

1.5.PREVISÃO DE MANUTENÇÃO

Prever o número de solicitações de mudança para um sistema requer uma compreensão do relacionamento entre o sistema e seu ambiente externo. Para avaliar os relacionamentos entre um sistema e seu ambiente, deve-se avaliar:

 O número e o complexidade dos interfaces de sistema.

 O número de requisitos inerentemente voláteis de sistema.

 Os processos de negócio em que o sistema é usado.

1.6.REENGENHARIA DE SOFTWARE

Existem dois benefícios importantes na reengenharia:

 Risco reduzido

 Custo reduzido

1.7.PROCESSO DE REENGENHARIA

TESTE DE SOFTWARE

É o acto de verificar se um software cumpre com os requisitos traçados. Essa verificação consiste na descoberta de erros, anomalias, ou informações sobre os atributos não funcionais do software.
O processo de teste tem 2 **objectivos**:

 Demonstrar ao desenvolvedor e ao cliente que o software atende a seus requisitos.

 Descobrir situações em que o software se comporta de maneira incorreta, indesejável ou de forma diferérite das especificações

Os testes mostram apenas a presença de erros,e não a sua ausência.
O teste é parte de um amplo processo de verificação e validação (V&V).
Barry Boehm, expressou sucintamente a diferença entre **validação** e **verificação**:

 Validação: estamos construindo o produto certo?

 Verificação: estamos construindo o produto da maneira certa?

2.1.Estágios de Teste

 Testes em desenvolvimento – testa-se durante o desenvolvimento para descobrir bugs.

 Teste de release – testa-se a versão completa do sistema antes de entrar em produção.

 Testes de utilizador – testa-se a versão inicial do sistema pelos potencias usuários

Durante o desenvolvimento, o teste pode ocorrer em **três níveis de granulandade**:

Testeunitário, em que as unidades individuais de programa ou classes de objetos são testadas individualmente.Testes unitários devem centrar-se em testar a funcionalidade dos objetos ou métodos.

Testede componentes, em que várias unidades indíviduais são integradas para criar componentes compostos. Testes de componentes devem centrar-se em testar as interfaces dos componentes.

Testede sistema, em que alguns ou todos os componentes de um sistema estão integrados e o sistema é testado como um todo. O teste de sistema deve centrar-se em testar asinterações entre os componentes.