Universidade Federal
de São João del-Rei

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
INTRODUÇÃO À ENGENHARIA DE SOFTWARE

DICIONÁRIO

Matheus N Silva

São João del-Rei
2023

Sumário

1 INTRODUÇÃO

Esse arquivo contem palavras e informações pertinentes à disciplina de Introdução à engenharia de software.

2 DICIONÁRIO

2.1 Capitulo 1

2.1.1 Engenharia de Software

A engenharia de software se destina a apoiar o desenvolvimento de software profissional em vez de a programação individual. Ela inclui técnicas que apoiam a especificação, o projeto e a evolução do software.

2.1.2 Software

São programas de computador e documentação associada.

2.1.3 Aplicações stand-alone

São sistemas de aplicação executados em um computador pessoal ou aplicativos que rodam em dispositivos móveis. Elas incluem toda a funcionalidade necessária e podem não necessitar de conexão a uma rede. Como exemplos, temos as aplicações de escritório em um computador pessoal, programas de CAD. software de manipulação de imagens, aplicativos de viagem, aplicativos de produtividade etc.

2.1.4 Aplicações interativas baseadas em transações

São aplicações executadas em um computador remoto e que são acessadas por usuários a partir de seus próprios computadores, smartphones ou tablets. Elas incluem aplicações web como as de comércio eletrônico, por exemplo, por meio das quais se interage com um sistema remoto para comprar bens e serviços. Essa classe de aplicação também inclui sistemas de negócio, nos quais uma empresa concede acesso a seus sistemas por meio de um navegador, de um programa cliente de uso específico ou de um serviço baseado na nuvem, como e-mail e compartilhamento de imagens. As aplicações interativas incorporam frequentemente um grande armazenamento de dados que é acessado e atualizado em cada transação.

2.1.5 Sistemas de controle embarcados

São sistemas de controle de software que controlam e gerenciam dispositivos de hardware. Em números, existem provavelmente mais sistemas embarcados do que qualquer outro tipo. Exemplos de sistemas embarcados incluem o software de um telefone celular, o software que controla o freio ABS em um carro e o software em um forno de micro-ondas para controlar o processo de cozimento.

2.1.6 Sistemas de processamento em lotes (batch)

São sistemas de negócio concebidos para processar dados em grandes lotes. Eles processam números enormes de entradas individuais para criar as saídas correspondentes. Exemplos de sistemas em lote incluem o faturamento periódico, como as contas de telefone, e os sistemas de folha de pagamento.

2.1.7 Sistemas de entretenimento

São destinados para uso pessoal, para entreter o usuário. A maioria desses sistemas consiste em jogos de gêneros variados, que podem ser executados em um console concebido especificamente para essa finalidade. A qualidade da interação com o usuário é a característica diferenciadora mais importante dos sistemas de entretenimento.

2.1.8 Sistemas para modelagem e simulação

São desenvolvidos por cientistas e engenheiros para modelar processos físicos ou situações que incluem muitos objetos diferentes e que interagem. Costumam ser computacionalmente intensivos e demandam sistemas paralelos de alto desempenho para a sua execução.

2.1.9 Sistemas de coleta de dados e análise

São aqueles que fazem a sua coleta no ambiente e enviam esses dados para outros sistemas, para processamento. O software pode ter de interagir com sensores e frequentemente é instalado em um ambiente hostil, como o interior de um motor, ou em uma localização remota. A análise de 'Big Data (grandes volumes de dados) pode envolver sistemas baseados na nuvem executando análises estatísticas e procurando relações entre os dados coletados.

2.1.10 Sistemas de sistemas

São utilizados em empresas e outras grandes organizações e são compostos de uma série de outros sistemas de software. Alguns deles podem ser produtos de software genéricos, como um sistema ERP Outros sistemas do conjunto podem ser desenvolvidos especialmente para esse ambiente

2.2 Capitulo 2

2.2.1 Processos de software

Os processos de software são sequências intercaladas de atividades técnicas, colaborativas e gerenciais, cujo objetivo global é especificar, projetar, implementar e testar um sistema de software.

2.2.2 Modelo cascata

Esse modelo considera as atividades fundamentais do processo de especificação, desenvolvimento,validação e evolução, e representa cada uma delas como fases distintas como: especificação de requisitos, projeto de software, implementação, testes e assim por diante

2.2.3 Desenvolvimento incremental

Essa abordagem intercala as atividades de especificação, desenvolvimento e validação. O sistema é desenvolvido como uma série de versões (incrementos), de maneira que cada versão adiciona funcionalidade à anterior

2.2.4 Engenharia de software orientada a reuso

Essa abordagem é baseada na existência de um numero significativo de componentes reutilizáveis. O processo de desenvolvimento do sistema concentra-se na integração desses compotentes em um sistema ja existente, em vez de desenvolver um sistema a partir do zero.

2.2.5 Rational Unified Process - RUP (Modelo de processo unificado)

Consiste em um modelo de processo interativo e incremental baseado na Linguagem de Modelagem Unificada (UML em ingles) e na programação orientada à objetos, mas

não se limita a isso.

2.3 Capitulo 3

2.3.1 Manifesto ágil

O principio do manifesto ágil consiste em: Envolvimento do cliente, acolher mudanças, entrega incremental, manter a simplicidade, pessoas, não processos

2.3.2 Extreme programming

O XP foi criado na década de 90 por Kent Beck, Ward Cunningham e Ron Jeffries, e ainda é um método bastante utilizado, um dos quais foca na satisfação do cliente e na entrega incremental, possuindo os objetivos de desenvolver sistemas rápidos e corretos, priorizando principalmente o desenvolvimento do software sobre a análise e o projeto do sistema.

2.3.3 Refatoração

A refatoração de software é o processo de melhorar o código de um sistema por meio de técnicas avançadas de programação. É o retrabalho de um código legado para otimizar a estrutura, organização e compreensão do código fonte.

2.3.4 Test-first

a escrita de testes primeiro define implicitamente tanto uma interface como uma especificação do comportamento para a funcionalidade que está sendo desenvolvida.

2.3.5 Programação em pares

Na programação os pares são criados dinamicamente para que todos os membros do time trabalhem uns com os outros durante o processo de desenvolvimento. A programação em pares tem uma serie de vantagens:

- Apoia a ideia de propriedade e responsabilidade coletivas pelo sistema.
- Ela age como um processo de revisão informal, já que cada linha de código é examinada por ao menos duas pessoas.

- Incentiva a refatoração para melhorar a estrutura do software.

2.4 Capitulo 4

2.4.1 Requisitos funcionais

São declarações dos serviços que o sistema deve fornecer, do modo como o sistema deve reagira determinadas entradas e de como deve se comportarem determinadas situações.

2.4.2 Requisitos não funcionais

São restrições sobre os serviços ou funções oferecidas pelo sistema. Os requisitos não funcionais se aplicam, frequentemente, ao sistema como um todo, em vez de às características individuais ou aos serviços.

2.4.3 Especificação de requisitos

O documento de requisitos de software, às vezes chamado Especificação de Requisitos de Software (SRS - do inglês Software Requirements Specification), é um a declaração oficial de o que os desenvolvedores do sistema devem implementar. Deve incluir tanto os requisitos de usuário para um sistema quanto uma especificação detalhada dos requisitos de sistema.

2.4.4 Stakeholders

Stakeholders significa público estratégico e descreve todas as pessoas ou "grupo de interesse"que são impactados pelas ações de um empreendimento, projeto, empresa ou negócio.

2.4.5 Estória

Estórias (Histórias e cenários) descrevem o que as pessoas fazem, quais informações usam e produzem e quais sistemas podem adotar nesse processo.

2.5 Capitulo 5

2.5.1 Modelos de contexto

Em um estágio inicial da especificação de um sistema, você deve decidir os limites do sistema. Isso envolve trabalhar com os stakeholders do sistema para decidir qual funcionalidade deve ser incluída no sistema e o que é fornecido pelo ambiente do sistema.

2.5.2 Modelos de interação

Todos os sistemas envolvem algum tipo de interação. Pode-se ter interação do usuário, que envolve entradas e saídas, interação entre o sistema que está em desenvolvimento e outros sistemas, ou interação entre os componentes do sistema.

2.5.3 Modelos estruturais

Os modelos estruturais de softwares exibem a organização de um sistema em termos de seus componentes e seus relacionamentos. Os modelos estruturais podem ser modelos estáticos, que mostram a estrutura do projeto do sistema, ou modelos dinâmicos, que mostram a organização do sistema quando ele está em execução

2.5.4 Modelos comportamentais

Modelos comportamentais são modelos do comportamento dinâmico do sistema quando está em execução. Eles mostram o que acontece ou deve acontecer quando o sistema responde a um estímulo de seu ambiente. Você pode pensar nesse estímulo como sendo de dois tipos:

- Dados - alguns dados que chegam precisam ser processados pelo sistema.
- Eventos - alguns eventos que acontecem disparam o processamento do sistema. Eles podem ter dados associados, mas nem sempre esse é o caso

2.5.5 Engenharia dirigida a modelos

A engenharia dirigida por modelos (MDE) é uma abordagem para o desenvolvimento na qual os modelos são as saídas principais do processo de desenvolvimento.

2.6 Capitulo 6

2.6.1 Comunicação de stakeholders

A arquitetura é uma apresentação de alto nível do sistema e pode ser usada como um foco de discussão por uma série de diferentes stakeholders.

2.6.2 O padrão MVC(Modelo-Visão-Controlador)

Separa a apresentação e a interação dos dados do sistema. O sistema é estruturado em três componentes lógicos que interagem entre si. O componente Modelo gerência os dados do sistema e as operações a eles associadas. O componente Visão define e gerência como os dados são apresentados ao usuário. O componente Controlador gerência a interação do usuário (por exemplo, pressionamento de teclas, cliques de mouse etc.) e passa essas interações para Visão e Modelo.

2.6.3 O padrão de arquitetura em camadas

Organiza o sistema em camadas, com funcionalidade associada a cada uma. Uma camada fornece serviços para a camada acima dela, então as camadas nos níveis mais inferiores representam os serviços essenciais que tendem a ser utilizados em todo o sistema.

2.7 Capitulo 7

2.7.1 UML - Unified Modeling Language

UML foi criada para estabelecer uma linguagem visual comum no complexo mundo do desenvolvimento de software, que também poderia ser compreendida por usuários do mundo dos negócios e qualquer pessoa que queira entender mais sobre um sistema.

2.7.2 Padrão de projetos

O padrão de projetos é uma descrição do problema e a essência de sua solução, de modo que ela possa ser reutilizada em diferentes contextos. O padrão de projetos não é uma especificação detalhada. Em vez disso, é possível encará-lo como uma descrição da sabedoria e do conhecimento acumulados, uma solução testada e aprovada para um problema comum.

2.7.3 Open source

Open source são programas ou apps distribuídos com código-fonte, o que permite que qualquer pessoa com conhecimentos de programação use, modifique, aprimore e até mesmo os compartilhe na internet.

2.8 Capitulo 8

2.8.1 Teste de desenvolvimento

O teste de desenvolvimento inclui todas as atividades de teste executadas pelo time responsável pelo sistema. O testador do software normalmente é o programador que o desenvolveu. Existem três estágios do teste de desenvolvimento:

- **Teste de unidade** - onde são testadas unidades de programa ou classes individuais. Esse tipo de teste deve se concentrar em testar a funcionalidade dos objetos e seus métodos.

- **Teste de componentes** - onde várias unidades são integradas, criando componentes compostos. Esse teste deve se concentrar em testar as interfaces dos componentes que promovem acesso às suas funções.

- **Teste de sistema** - onde alguns ou todos os componentes em um sistema são integrados e o sistema é testado como um todo. O teste de sistema deve se concentrar em testar as interações dos componentes.

2.8.2 Desenvolvimento dirigido por testes

O desenvolvimento dirigido por testes (TDD. do inglês test-driven development) é uma abordagem para desenvolvimento de programas na qual se intercalam testes e desenvolvimento do código (BECK. 2002; JEFFRIES; MELNIK. 2007). Esse código é desenvolvido incrementalmente, junto a um conjunto de testes para cada incremento, e o próximo incremento não começa até que o código que está sendo desenvolvido seja aprovado em todos os testes. O desenvolvimento dirigido por testes foi introduzido como parte do método de desenvolvimento ágil Programação Teste de software Extrema (XP). Hoje, no entanto, ganhou aceitação geral e pode ser utilizado tanto em processos ágeis como em processos dirigidos por plano.

2.8.3 Teste de lançamento

O teste de lançamento (release) é um processo de teste de um determinado lançamento de um sistema, destinado ao uso fora do time de desenvolvimento. Normalmente, o lançamento do sistema é voltado para clientes e usuários. Entretanto, em um projeto complexo, o lançamento poderia ser para outros times que estão desenvolvendo sistemas relacionados.

2.8.4 Teste de usuário

O teste de usuário é um estágio no processo de teste no qual os usuários ou clientes fornecem entradas e conselhos sobre o teste de sistema. Isso pode envolver o teste formal de um sistema que foi contratado de um fornecedor externo. Por outro lado, pode ser um processo informal em que os usuários experimentam um novo produto de software para ver se gostam e conferem se ele atende suas necessidades. O teste de usuário é essencial, mesmo em casos em que já tenham sido realizados testes abrangentes de sistema e de lançamento, pois as influências do ambiente de trabalho do usuário podem ter um efeito importante na confiabilidade, no desempenho, na usabilidade e na robustez de um sistema.

2.9 Capitulo 9

2.9.1 Processo de evolução

Assim como em todos os processos de software, não existe um processo padrão de mudança ou de evolução de software. O processo de evolução mais adequado para um sistema de software depende do tipo de software que está sendo mantido, dos processos de desenvolvimento de software utilizados em uma organização e das habilidades das pessoas envolvidas.

2.9.2 Manutenção de software

Existem três tipos diferentes de manutenção de software:

- Reparo de defeitos para corrigir bugs e vulnerabilidades.
- Adaptação ao ambiente para adaptar o software a novas plataformas e ambientes

- Acréscimo de funcionalidade para adicionar novas características e apoiar novos requisitos.

2.9.3 Reengenaria de software

A reengenharia pode envolver a redocumentação de sistema, a refatoração da arquitetura de sistema, a mudança de linguagem de programação para uma linguagem moderna e modificações e atualizações da estrutura e dos dados de sistema.

2.9.4 Sistemas legados

Sistemas legados são plataformas ou softwares que estão ultrapassados. Isso quer dizer que, com os avanços da tecnologia, essa infraestrutura começa a se tornar um problema na dinâmica da empresa. Assim, o sistema antigo não oferece clareza e transparência nas operações com um banco de dados e funcionalidades desatualizados

Referências

[ALBERGARA, 2023] ALBERGARA, E. T. (2023). *Introdução à engenharia de software.*

[SOMMERVILLE, 2018] SOMMERVILLE, I. (2018). *Engenharia de Software.* Pearson.