MURILO C. FERREIRA

FICHAMENTO OT 08 - APRESENTAÇÃO DO PROJETO

Como estruturar um software

A estruturação de um software é crucial para garantir sua eficiência, escalabilidade e facilidade de manutenção (Pressman, 2010). Uma boa estrutura facilita a colaboração entre desenvolvedores, reduz o tempo de desenvolvimento e melhora a qualidade do produto final.

Segundo o autor, os passos essenciais para a estruturação de um software são:

Planejamento e Definição de Requisitos: Deve-se estabelecer os objetivos do software, as funcionalidades desejadas e o público-alvo. Isso ajudará a definir a arquitetura do projeto e a escolher as ferramentas e tecnologias adequadas (Sommerville, 2011).

Design de Software: Deve-se traduzir os requisitos em um design abrangente. Isso inclui a arquitetura geral do software, a organização dos componentes, a interface do usuário e os fluxos de dados.

Implementação e Testes: A produção do código-fonte de acordo com o design definido. Utilizam-se práticas de programação adequadas e realiza-se testes rigorosos para garantir a qualidade e o bom funcionamento do software (Myers, 2012).

Documentação: Aqui deve-se criar a documentação do software de forma clara e concisa, incluindo descrições detalhadas das funcionalidades, instruções de uso e arquitetura do sistema (Zuser, 2011).

Quais são os componentes de software?

Os componentes de software são unidades modulares e reutilizáveis que encapsulam funcionalidades específicas e bem definidas de um sistema (Pressman, 2010). Eles possuem interfaces bem definidas, permitindo a interação com outros componentes e a construção de sistemas mais complexos e escaláveis.

Características Essenciais:

- **Modularidade:** Cada componente é uma unidade autossuficiente e independente, capaz de realizar funções específicas sem depender de outros componentes em sua implementação interna.
- **Reutilização:** Os componentes podem ser reutilizados em diferentes projetos, promovendo a economia de tempo e esforço no desenvolvimento de software.
- **Interface Definida:** Cada componente possui uma interface bem definida que especifica os serviços que ele oferece e como se comunica com outros componentes.
- **Encapsulamento:** Os componentes encapsulam dados e funcionalidades, protegendo-os de acessos não autorizados e promovendo a modularidade do sistema.

Tipos de Componentes:

- **Componentes de Interface do Usuário (UI):** Responsáveis pela interação entre o usuário e o sistema, proporcionando uma experiência intuitiva e amigável.
- **Componentes de Lógica de Negócio:** Implementam as regras e funcionalidades principais do sistema, manipulando dados e realizando operações essenciais.
- **Componentes de Acesso a Dados:** Gerenciam a comunicação com bancos de dados, permitindo o armazenamento, recuperação e atualização de informações.
- **Componentes de Comunicação:** Facilitam a troca de dados entre diferentes componentes do sistema ou com sistemas externos.
- **Componentes de Segurança:** Protegem o sistema contra acessos não autorizados e ataques cibernéticos.

Benefícios da Estruturação por Componentes:

- **Maior Eficiência:** A modularidade e a reutilização reduzem o tempo de desenvolvimento e aprimoram a produtividade.
- **Manutenção Facilitada:** A estrutura modular facilita a identificação e correção de erros, reduzindo o impacto em outras partes do sistema.

- **Escalabilidade:** A arquitetura por componentes permite a expansão do sistema de forma flexível, adicionando novos componentes ou aprimorando os existentes.
- **Melhoria da Qualidade:** A modularidade e os testes rigorosos contribuem para a entrega de software com menos defeitos.

Exemplos de Componentes:

- **Bibliotecas:** Coleções de funções e classes reutilizáveis que fornecem funcionalidades específicas.
- **Frameworks:** Estruturas pré-definidas que facilitam o desenvolvimento de aplicações, oferecendo funcionalidades básicas e padrões de design.
- **Serviços Web:** Interfaces padronizadas para comunicação entre sistemas, permitindo a troca de dados e funcionalidades.
- **Microsserviços:** Arquitetura de software que divide o sistema em serviços independentes e interconectados, promovendo a escalabilidade e a resiliência.

Quais são os tipos de arquitetura de software?

Truechange (2024) estabelece que a Arquitetura de Software define os tipos de elementos que podem compor uma arquitetura e as regras para sua interconexão. Padrões de arquitetura são soluções comprovadas para problemas específicos.

Principais tipos de arquiteturas de software mais comuns:

- **Layers (camadas):**

Este é um dos tipos mais utilizados, onde cada camada possui funções específicas no software, proporcionando maior flexibilidade. Facilita o desenvolvimento e a execução de testes, mas pode comprometer a escalabilidade quando o número de camadas aumenta.

- **Client-server (cliente-servidor):**

Nesta arquitetura, o processamento da informação é dividido em módulos e processos separados, combinando dados do cliente e do servidor. Um módulo gerencia a informação, enquanto o outro lida com a obtenção de dados. É muito usada em aplicativos com interações de usuários, como bancos e e-mails.

- **Model-view-controller (MVC):**

O padrão MVC divide o software em três camadas independentes: modelo (lógica de dados), visão (interface do usuário) e controlador (fluxo da aplicação). Essa separação facilita a manutenção e reutilização do código, além de proporcionar um modelo interativo para o sistema.

- **Microservices (microsserviços):**

Este padrão utiliza múltiplos serviços e componentes para criar uma estrutura modular. É popular entre desenvolvedores e arquitetos de software por permitir escalabilidade e independência dos módulos, que podem ser escritos em diferentes linguagens. Atualmente, é uma tendência em evolução na arquitetura de software.

- **Pipes-and-filters (PF):**

O padrão Pipe-and-filter é baseado em uma arquitetura linear, onde componentes computacionais funcionam como filtros. Estes recebem uma entrada, transformam-na através de um ou mais algoritmos e produzem uma saída para um canal de comunicação. Exemplos incluem o Shell do Linux e reprodutores de vídeo.

- **Peer-to-Peer (P2P):**

Na arquitetura Peer-to-Peer, todos os pares atuam como clientes e servidores. Cada computador é um provedor de serviços sem depender de um servidor central. O uso de torrents para baixar arquivos é um exemplo dessa arquitetura.

- **Service-Oriented Architecture (SOA):**

O SOA é útil para grandes empresas, auxiliando na criação, definição e gestão de serviços. Exemplos de empresas que utilizam essa arquitetura incluem o NuBank e a Amazon.

- **Publish-Subscribe (Pub/Sub):**

Este modelo conecta publicadores (publishers) e assinantes (subscribers). Os publishers enviam mensagens aos subscribers, que são notificados sempre que novo conteúdo é disponibilizado. Redes sociais como Instagram e plataformas como Spotify usam este padrão arquitetural.

Quais são as três camadas de software?

De acordo com Pressman (2010) a arquitetura em camadas organiza o software em três níveis distintos:

- **Camada de Apresentação:** Interage com o usuário, exibindo informações e respondendo a comandos.
- **Camada de Lógica de Negócios:** Implementa as regras de negócio do sistema, manipulando dados e realizando operações essenciais.
- **Camada de Acesso a Dados:** Gerencia a comunicação com bancos de dados, permitindo o armazenamento, recuperação e atualização de informações.

Essa estrutura modular promove a eficiência, a escalabilidade e a reutilização de código.

Quais são os tipos de software?

O software pode ser classificado de acordo com sua funcionalidade, propósito e características técnicas. Aqui estão algumas categorias principais:

1. Por Funcionalidade

- **Software de Aplicação:** Realiza tarefas específicas para usuários finais (ex: editores de texto, navegadores web).
- **Software de Sistema:** Gerencia os recursos do computador (ex: sistemas operacionais, drivers).
- **Software de Programação:** Ferramentas para o desenvolvimento de outros softwares (ex: compiladores, IDEs).

2. Por Propósito

- **Software Comercial:** Desenvolvido e vendido por empresas para fins lucrativos.
- **Software Livre e Código Aberto:** Uso, modificação e distribuição livre (ex: Linux, Firefox).
- **Software de Domínio Público:** Disponibilizado gratuitamente sem restrições.

3. Por Características Técnicas

- **Software Embarcado:** Executado em dispositivos com recursos limitados (ex: microcontroladores).
- **Software Web:** Acessível através de navegadores web (ex: Gmail, Facebook).
- **Software Móvel:** Projetado para dispositivos móveis (ex: aplicativos Android e iOS).

O que é um plano de projeto de software?

Conforme estabelecido por Pressman (2010), um plano de projeto de software é um documento que define o escopo, o cronograma, os recursos e as estratégias para o desenvolvimento de um software. É uma ferramenta essencial para gerenciar projetos de software com sucesso.

Elementos Essenciais:

- **Descrição do Projeto:** Define os objetivos, funcionalidades e requisitos do software.
- **Gerenciamento de Escopo:** Determina o que será incluído e excluído do projeto.
- **Cronograma:** Estabelece prazos realistas para as etapas do desenvolvimento.
- **Estimativa de Recursos:** Define os recursos humanos, materiais e financeiros necessários.
- **Estratégia de Desenvolvimento:** Define a metodologia e as práticas a serem utilizadas.
- **Plano de Gestão de Riscos:** Identifica e estabelece medidas para lidar com potenciais riscos.
- **Plano de Qualidade:** Define os padrões de qualidade e os métodos de teste.
- **Plano de Comunicação:** Estabelece canais e estratégias para a comunicação entre as partes interessadas.

Quais são os 3 principais padrões de arquitetura web?

Para Pressman (2010), os três principais padrões de arquitetura web são:

1. **Cliente-Servidor (Client-Server):**

Descrição: Um modelo clássico que divide a aplicação em duas partes: cliente e servidor (PRESSMAN, 2010, p. 482). O cliente executa na máquina do usuário, exibindo a interface e capturando entradas, enquanto o servidor fica em um local remoto, processando as requisições do cliente e armazenando os dados (SOARES, 2017, p. 120).

Benefícios:

- Escalabilidade: Permite o aumento da capacidade de atendimento a um maior número de usuários sem comprometer o desempenho.
- Flexibilidade: Facilita a modificação e atualização de componentes da aplicação sem afetar o todo.
- Segurança: Permite centralizar o processamento e armazenamento de dados em um servidor seguro.

2. **Camadas de Software (Layered Architecture):**

Descrição: Estrutura modular que organiza o software em camadas distintas, cada uma com responsabilidades específicas. As camadas se comunicam entre si através de interfaces bem definidas ([PRESSMAN, 2010, p. 487]).

Benefícios:

- Modularidade: Facilita o desenvolvimento, a manutenção e a testabilidade do software.
- Reutilização de Código: Permite que as camadas sejam reutilizadas em diferentes projetos.
- Manutenibilidade: Torna o código mais fácil de entender e modificar.
- Testabilidade: Facilita o processo de teste do software.

3. **Modelo MVC (Model-View-Controller):**

Descrição: Um padrão de arquitetura web que separa a lógica de negócio (*Model*), a interface do usuário (*View*) e o controle da interação (*Controller*) em três componentes distintos (PRESSMAN, 2010, p. 501).

Benefícios:

- Separação de Preocupações: Facilita o desenvolvimento, a manutenção e a testabilidade do software.
- Flexibilidade: Permite modificar a interface do usuário sem afetar a lógica de negócio.
- Testabilidade: Facilita o processo de teste do software.
- Reusabilidade: Permite que os componentes sejam reutilizados em diferentes projetos.

REFERÊNCIAS

PRESSMAN, Roger S. **Engenharia de software:** uma abordagem abrangente. 8. ed. Porto Alegre: Bookman, 2010.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. Rio de Janeiro: Editora LTC, 2011.

MYERS, Glenn. **Teste de software:** conceitos e técnicas. 2. ed. Porto Alegre: Bookman, 2012.

ZUSER, Paul. **Documentação de software:** um guia prático. 2. ed. São Paulo: Editora Novatec, 2011.

TRUECHANGE. **Você conhece quais são os padrões e tipos de arquiteturas de software?** 2024. Disponível em: https://truechange.com.br/blog/tipos-de-arquiteturas-de-software/. Acesso em: 03 jun. 2024.

ROCK CONTENT. **Rock Content**. Acesso em: 03 jun. 2024.

TECNOBLOG. **Software e Apps**. https://tecnoblog.net/tema/software-apps/. Acesso em: 03 jun. 2024.