PUC
CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS
PADRÕES E ARQUITETURA DE SOFTWARE

Jéssica Silva Kushida - 23012069
Marcela de Siqueira Franco - 23013238
Natália Naomi Sumida - 23008255
Nicole Silvestrini Garrio - 23009486

CAMPINAS
2025

SUMÁRIO

1. Objetivo da atividade

Nesta atividade, foi proposta a leitura do Módulo 1 do livro ‘Arquitetura de Software’, do autor Giocondo Marino Antonio Gallotti. A partir da leitura, deve ser feito um breve resumo das arquiteturas descritas no módulo.

2. Conceitos de Software

Neste primeiro tópico de Conceitos Arquiteturais, o autor introduz o assunto com a seguinte pergunta ao leitor: ‘o que é um software?’, pois, para entendermos como é a sua arquitetura, primeiro devemos compreender a sua definição.
O software apresenta-se como um elemento intermediário entre o ser humano e a máquina. Essa intermediação é feita por meio do processamento de dados, que podem ser de qualquer origem, desde que esses dados possam ser convertidos para um formato que o computador consiga entender.
Além disso, a interação entre o usuário e o software ocorre através da chamada ‘interface homem-máquina’. Porém, o software que usamos, como um editor de texto, não “conversa” diretamente com o processador do computador. Essa comunicação é feita com o sistema operacional, que é o responsável por traduzir as informações recebidas, convertendo à linguagem de máquina, que assim o computador é capaz de entender.
Por último, é pontuado o funcionamento do software em si, que é a execução de algoritmos, esses que são séries de instruções que, para que uma tarefa seja atendida, devem seguir uma sequência correta.

3. Função e Desempenho

Para que possamos entender a organização de um software, podemos pensar que ele é dividido em dois elementos: função e desempenho. Estes elementos estão presentes desde o início de desenvolvimento do software e estão diretamente relacionados, mas essa relação pode variar dependendo do caso.
A função é a forma como o software manipula os dados, geralmente seguindo processos. Na parte da função, o desempenho não é crucial, apesar de ser importante que os processos aconteçam rapidamente. Por outro lado, ao tratar-se de softwares que controlam vários processos simultaneamente, o desempenho é um fator essencial no funcionamento dos programas.

Portanto, devemos buscar manter o equilíbrio entre função e desempenho, através de escolher e criar componentes que ajudem durante o desenvolvimento.

4. Fases do Desenvolvimento do Software

Neste tópico, para entendermos o desenvolvimento de software, ele é dividido em três partes, são elas: definição; desenvolvimento; verificação, liberação e manutenção.

4.1 Definição

Nessa fase, o principal objetivo é planejar o desenvolvimento antes de qualquer codificação. Essa etapa inicia-se com a criação de um plano de projeto, no qual será definido o propósito do software, problemas que ele deve resolver, suas soluções e quais necessidades ele atenderá, destacando casos de encomendas.

Ademais, uma análise de riscos é feita, considerando a possibilidade de riscos durante o desenvolvimento. Também é feito um levantamento de recursos, como equipe, prazos, custos e materiais. Essa parte é essencial para avaliar a viabilidade e compensação do projeto a ser executado.

Em seguida, ocorre o levantamento de requisitos, onde é detalhado as funcionalidades do software com a ajuda de protótipos, diagramas e pseudocódigos. Nessa fase, o cliente pode participar com o intuito de garantir que suas necessidades estão sendo compreendidas corretamente.

Por último, o documento de especificação de requisitos do software é criado, no qual a definição do que o sistema deve ou não deve dispor. Esse documento é revisado pelos envolvidos no projeto e servirá de base para a próxima fase.

4.2 Desenvolvimento

Na fase de desenvolvimento, o trabalho começa na análise dos documentos preparados na fase de definição. Com base nisso, é dado início à criação do algoritmo, a definição da estrutura do sistema e interfaces, sempre levando como referência o plano de projeto a fim de garantir a qualidade do que é feito. Tudo que é construído é documentado em uma especificação de projeto, onde são descritas as configurações do software e os procedimentos adotados para cada módulo.

Em seguida, é feita a codificação, ou seja, a etapa na qual o programa é escrito utilizando uma linguagem de programação ou ferramentas CASE. Destaca-se também o fato de que um bom código depende de um bom projeto, por isso ele deve ser claro, objetivo e bem estruturado para garantir concordância com as especificações determinadas. Por fim, é gerada a linguagem-fonte do sistema, separada por módulos.

4.3 Verificação, liberação e manutenção

Nesta última fase, o software passa por testes rigorosos com o objetivo de identificar erros de funcionalidade e de desempenho. Os testes primeiramente são feitos nos módulos separadamente, depois, na integração entre eles, e por último é feita uma validação geral do sistema. Em caso de falhas, é feito o debugging para correção.

Antes da liberação, ocorre uma checagem de qualidade que confere se toda a documentação está correta. E finalmente o software é entregue aos usuários finais, mas o trabalho continua através da manutenção do sistema, corrigindo problemas encontrados, adaptando e melhorando o produto.

5. Componentes e Conectores

O tópico introduz a explicação de que a arquitetura de software é a estrutura interna de um sistema, ou seja, é a forma como o sistema se organiza e funciona. Porém, essa arquitetura dita não é estática, ela pode evoluir e mudar durante o desenvolvimento, seja por feedbacks, novas funções e correções.

O objetivo é que, além de ser clara e simples, a arquitetura tenha quatro características fundamentais:

- Flexibilidade: pois permite fazer alterações, melhorias e correções sem a necessidade de refazer grandes partes do projeto.
- Extensibilidade: para facilitar a incorporação de novos elementos e novas features.
- Portabilidade: execução do sistema em diferentes plataformas.
- Reutilização: permite usar a arquitetura desenvolvida em outros softwares.

O autor apresenta um famoso modelo de arquitetura de software, proposto por Dewayne E. Perry e Alexander L. Wolf, chamado ‘Fundamentos para o estudo da arquitetura de software’, onde é expressa a seguinte fórmula: ‘Arquitetura = (Elementos + Organização + Decisões)’. Esses ‘elementos’ podem ser divididos em três tipos:

- Elementos de processamento (operam os dados).
- Elementos de dados (matéria-prima a ser processada).
- Elementos de conexão (amarram e conectam os outros elementos para que tudo funcione em conjunto.

6. Configurações e Padrões Arquiteturais

Neste tópico, o autor pontua que, antes de conhecer os padrões arquiteturais, é importante entender que cada um deles possui suas características, suas vantagens

e desvantagens. Portanto, desenvolvedores precisam ter clareza sobre seus objetivos a fim de escolher o padrão mais adequado para seus projetos.
Um padrão arquitetural, de forma geral, descreve uma relação entre organização e elementos que já deram certo em projetos finalizados. Ele é descrito de forma abstrata, usando tabelas e diagramas.

6.1 Arquitetura MVC

Essa arquitetura é um padrão bastante presente em sistemas Web. Ele organiza o código de um sistema em três partes principais que se comunicam entre si. São os componentes: modelo (gerenciamento do sistema de dados e operações associadas), visão (como os dados são apresentados ao usuário) e controlador (gerenciamento da interação do usuário).

Ele é usado principalmente em sistemas onde existem múltiplas formas de visualizar e interagir com os mesmos dados.

Como vantagens, é possível citar a independência entre os componentes, pois os dados podem ser alterados sem afetar a forma como são exibidos.

Já desvantagens, em um sistema no qual as interações e o modelo de dados são muito simples, a implementação do MVC pode adicionar uma complexidade desnecessária.

6.2 Arquitetura em camadas

Nessa arquitetura, o sistema é organizado em camadas, onde cada uma oferece serviços para a camada imediatamente acima dela. Sendo assim, as camadas mais baixas contém os serviços mais gerais e fundamentais, que podem ser utilizados por todo o sistema.

Ela é utilizada em algumas situações, como: construção de novas funcionalidades em sistemas já existentes; desenvolvimento distribuído em equipes, com responsabilidades separadas; proteção multinível.

As vantagens dessa arquitetura se destacam na confiabilidade (adição de recursos de segurança em cada camada) e o fato de que é possível substituir uma camada inteira sem afetar o resto do sistema.

Já em desvantagens, é possível citar a difícil separação das responsabilidades de cada camada, a violação de camadas que pode quebrar a organização do padrão, e o desempenho, pois uma solicitação precisará passar e ser interpretada por múltiplas camadas.

6.3 Arquitetura de repositório

Na arquitetura de repositório, todos os dados do sistema são armazenados e gerenciados em um repositório central, logo os componentes não interagem entre si, somente com o repositório.

Esse padrão é utilizado principalmente em sistemas que trabalham com grandes volumes de informações que precisam ser armazenadas por um longo tempo.

Suas vantagens destacam a independência dos componentes e o gerenciamento centralizado.

Suas desvantagens sinalizam que esse padrão possui um ponto único de falha, pois, se o repositório falhar, o sistema inteiro será afetado, a ineficiência na comunicação e a dificuldade de distribuição.

6.4 Arquitetura cliente-servidor

Nessa arquitetura, a funcionalidade do sistema está organizada em serviços, onde cada serviço é prestado por um servidor. Os clientes são os usuários dos serviços e acessam os servidores para uso.

Ela é utilizada quando há a necessidade de acessar os dados do banco por uma série de locais.

A principal vantagem desse modelo é a possibilidade de distribuição dos servidores através de uma rede.

Já como desvantagens é possível citar a imprevisibilidade por causa da dependência da rede, problemas de gerenciamento por diferentes organizações e o fato de que cada serviço é um ponto único de falha.

6.5 Arquitetura de duto e filtro

Por último, a arquitetura de duto e filtro é caracterizada pela organização do processamento de dados, o qual cada componente de processamento (filtro) realiza um tipo de transformação de dados, assim os dados fluem como em um duto de um componente para outro.

Esse padrão é usado comumente em aplicações de processamento de dados em que as entradas são processadas em etapas separadas para gerarem saídas relacionadas.

As vantagens dessa arquitetura são o reuso da transformação dos dados, que é de fácil entendimento.

As desvantagens citam que o formato para as transferências de dados tem de ser acordado, o que aumenta o overhead do sistema e pode impossibilitar reuso de transformações devido a estruturas incompatíveis de dados.