

Engenharia de Software

06 - Arquitetura de Software

Prof. Msc. Francisco de Assis de Souza Rodrigues

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

"Ensina-me bom juízo e conhecimento, pois creio nos teus mandamentos." **Salmos 119:66**

Pedir orientação a Deus e valorizar o aprendizado nos ajuda a crescer com sabedoria e discernimento.

🎯 Objetivos

- Entender o que é arquitetura de software e por que é importante
- Identificar as decisões arquiteturais principais que afetam requisitos não funcionais
- Conhecer diferentes visões para documentar arquitetura
- Reconhecer e aplicar padrões/estilos arquiteturais comuns
- Comparar arquiteturas típicas (camadas, cliente-servidor, microsserviços)
- Aplicar os conceitos construindo uma proposta de arquitetura para um projeto real

UniFAP

Introdução à Arquitetura de Software

Definição: arquitetura refere-se às estruturas fundamentais de um sistema de software, seus componentes, os relacionamentos entre eles, os estilos de interação, políticas e decisões de design de alto nível.

Papel da arquitetura: é a ponte entre os requisitos (especialmente não funcionais) e o design detalhado.

Níveis de arquitetura:

- **Architecture in the small**: arquitetura de programas individuais, decomposição em componentes menores.
- **Architecture in the large**: sistemas empresariais distribuídos, múltiplos subsistemas, várias máquinas, possivelmente múltiplas organizações.

UniFAP

Importância de projetar & documentar arquitetura

- **Comunicação com stakeholders:** arquiteturas bem documentadas servem como referência comum.
- **Análise de requisitos não funcionais:** desempenho, segurança, escalabilidade, manutenção etc. Decisões arquiteturais influenciam esses requisitos.
- **Reuso em larga escala:** arquiteturas genéricas ou famílias de produto.

Decisões Arquiteturais (Architectural Design Decisions)

O que são decisões arquiteturais: escolhas de alto nível que definem a estrutura do sistema, controlam dependências, comunicação, modularização, distribuição, estilo etc.

Exemplos de decisões que precisam ser tomadas:

- Qual arquitetura genérica ou modelo de aplicação usar? (aplicação web, sistema embarcado, etc.)

UniFAP

Decisões Arquiteturais (Architectural Design Decisions)

- Como será a distribuição do sistema (monolito, distribuído, em nuvem)?
- Que estilos arquiteturais são apropriados?
- Como decompor o sistema em módulos ou componentes?
- Estratégia de controle (quem chama quem, quem coordena, como orquestrar componentes)
- Como avaliar arquiteturas alternativas? Critérios de não funcional (desempenho, escalabilidade, segurança, manutenibilidade)
- Como documentar (notações, visões, diagramas)

UniFAP

Visões Arquiteturais (Architectural Views)

Motivação: um único diagrama nunca mostra tudo; diferentes visões ajudam diferentes públicos / preocupações.

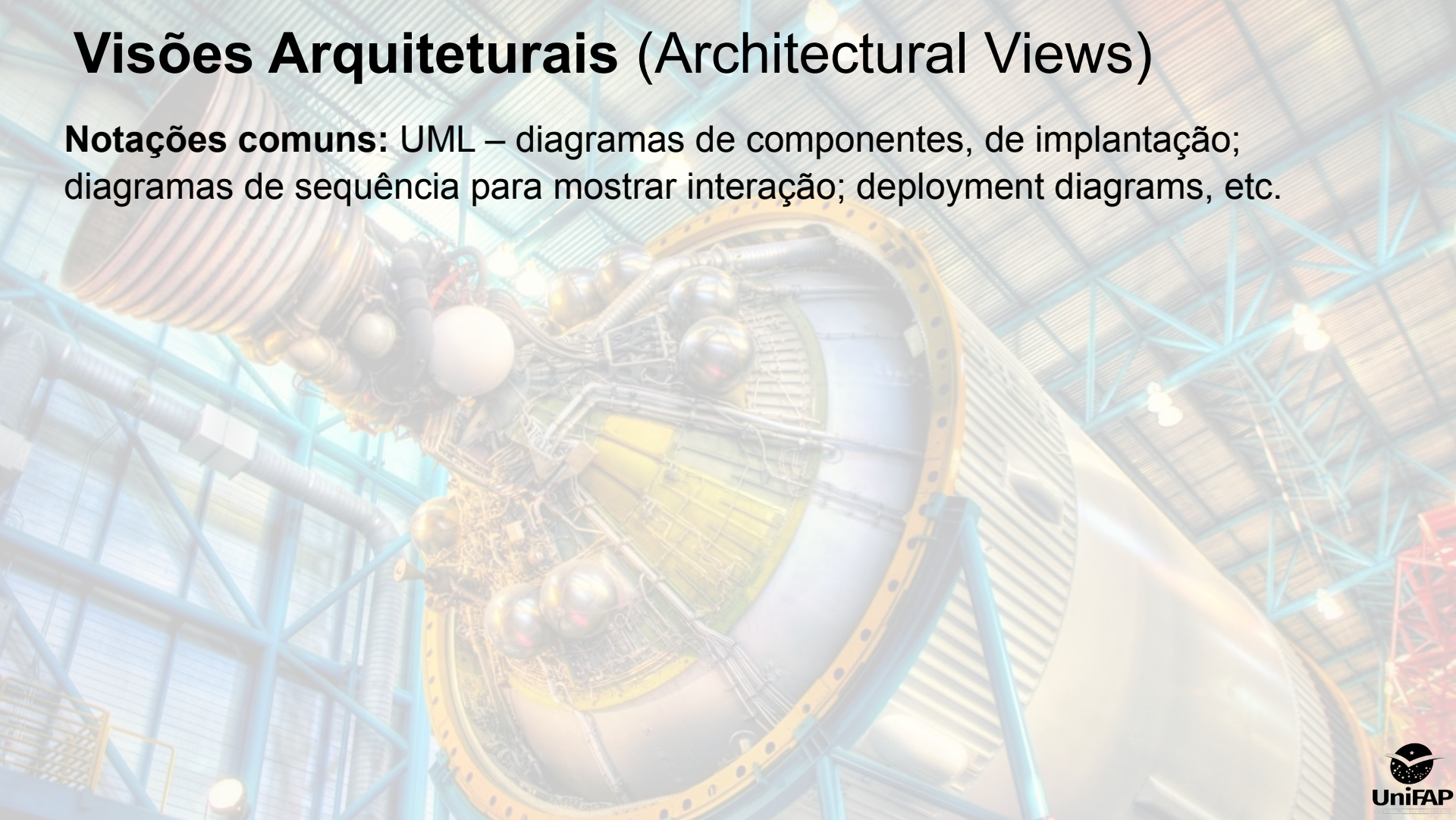
Principais visões segundo Sommerville:

Visão	Propósito / foco
Lógica (logical view)	abstrações principais, classes, modulos, dominando conceitualmente o sistema.
Processos (process view)	como os componentes interagem em tempo de execução, threads/processos, sincronização, concorrência.
Desenvolvimento (development view)	como o sistema está organizado para desenvolvimento: pacotes, módulos, versão, equipes.
Física (physical / deployment view)	mapeamento de software para hardware, máquinas, redes, comunicação entre nós.

UniFAP

Visões Arquiteturais (Architectural Views)

Notações comuns: UML – diagramas de componentes, de implantação; diagramas de sequência para mostrar interação; deployment diagrams, etc.



UniFAP

Padrões / Estilos Arquiteturais

O que é um padrão ou estilo arquitetural: description of generic architecture que foi testada em vários contextos; define componentes típicos, relacionamentos, vantagens e desvantagens.

UniFAP

Padrões / Estilos Arquiteturais

Exemplos frequentes:

- **Camadas (Layered Architecture / Abstract Machine / Layered Style)** – separar funcionalidades em camadas; define como camadas interagem; facilita manutenção, modularização.
- **Cliente-Servidor (Client-Server)** – clientes fazem requisições a servidores; servidores oferecem serviços.
- **Repositório / Repository style** – dados centralizados, muitos componentes acessam o repositório; bom para compartilhamento; porém pode gerar gargalos.

UniFAP

Padrões / Estilos Arquiteturais

Exemplos frequentes:

- **Pipe and Filter** – processos sequenciais com filtros; cada filtro processa dados e passa adiante.



Padrões / Estilos Arquiteturais

Exemplos frequentes:

- **MVC (Model-View-Controller)** – separar lógica de negócio (modelo), interface/presentação, controle de fluxo entre eles.



Vantagens vs desvantagens destes estilos (impacto em desempenho, complexidade, evolutibilidade, teste etc.)

UniFAP

Arquiteturas de Aplicação Genéricas

O que é: arquiteturas padrões para tipos comuns de aplicações (transações, processamento de dados, sistemas baseado em eventos, processamento de linguagens etc.).

Exemplos:

- Sistemas de processamento de dados (batch processing)
- Sistemas de transação interativo (por exemplo, sistemas de reservas, e-commerce)
- Sistemas de eventos (event-driven systems)
- Sistemas de linguagem / tradutores (compiladores, interpretadores)

Como essas arquiteturas seguem certos padrões ou estilos que combinam decisões: camadas + serviço + repositório etc.

UniFAP

Comparação prática de arquiteturas

Três arquiteturas a comparar:

Arquitetura em camadas

- **Estrutura típica:** apresentação, lógica de aplicação / negócio, persistência / dados, infra-estrutura – separação clara de responsabilidades.
- **Vantagens:** modularidade, facilidade de manutenção, isolamento, evolução controlada;
- **Desvantagens:** camadas podem introduzir overhead, latência, restrições de dependência.

UniFAP

Comparação prática de arquiteturas

Cliente-Servido (Client-Server)

- Clientes requisitam, servidores respondem; pode haver múltiplos servidores; distribuição;
- Vantagens: reutilização de servidores; escalabilidade; divisão clara;
- Desvantagens: falhas de servidor, latência, possíveis gargalos, complexidade de rede.

Comparação prática de arquiteturas

Microsserviços (Microservices)

- Cada serviço pequeno, autônomo, executa parte funcional; comunicação via APIs; possivelmente infraestrutura de serviços independentes;
- Vantagens: alta escalabilidade, equipes independentes, deploy independente, tolerância a falhas;
- Desafios: complexidade operacional, comunicação entre serviços, monitoramento, transações distribuídas, consistência, latência, overhead de rede.

UniFAP

Comparação prática de arquiteturas

Critérios de comparação relevantes:

- Escalabilidade
- Manutenibilidade / Evolução
- Desempenho / latência
- Complexidade de implantação e operação
- Resiliência / falhas
- Coesão e acoplamento

UniFAP

Estudo de caso prático

Tomamos como base o projeto que vocês estão desenvolvendo.

Proposta arquitetural: usar camadas, cliente-servidor ou microsserviços? Ou híbrida?

- Como ficaria o esboço: componentes, serviços, módulos, comunicação, deploy, base de dados etc.

Estudo de caso prático

Justifique as decisões:

- Quais decisões arquiteturais você está fazendo?
- Quais visões você vai documentar?
- Que padrões vai aplicar?
- Por que escolhas de arquitetura (camadas / cliente-servidor / microsserviços) são mais adequadas para aquele projeto, considerando os requisitos não funcionais?

UniFAP

Resumo / Takeaways

- Arquitetura define as estruturas principais de um sistema, é crucial para não funcionais
- Decisões arquiteturais devem ser feitas cedo, com consciência dos trade-offs
- Documentar diferentes visões ajuda stakeholders distintos e facilita manutenção / evolução
- Padrões arquiteturais dão “mapas” prontos, mas não são solução única: adaptá-los à realidade do projeto
- Para projeto real, avaliar bem o tipo de aplicação, requisitos, equipe, operação etc. para escolher entre camadas, cliente-servidor, microsserviços

Perguntas / Discussão

- Para que tipo de sistema microsserviços não é uma boa escolha?
- Como garantir que arquiteturas em camadas não se tornem rígidas demais?
- Que visões arquiteturais seriam mais importantes para determinados stakeholders (cliente, equipe de operação, equipe de desenvolvimento)?
- Quais padrões poderiam se combinar no projeto de vocês?

UniFAP

Bibliografia

SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo, SP: Pearson, 2018. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 15 ago 2025.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/168127



Referências

CHATGPT. Expansão conceitual sobre integração entre COBIT, ITIL e PMBOK. OpenAI, 2025. Disponível em: https://chat.openai.com/. Acesso em: 11 ago. 2025.


UniFAP


OBRIGADO
Arigato
Spasibo
Obrigado
Dankon
Maake
Multumesc
Ua Tsaug Rau Koj
Mochchakkeram
Kiitos
Merci
Chokrane
Terma Kasih
Grazie
Asante
Mamana
Kia Ora
THANK YOU
Raibh Maith Agat
Jusparxar
Welalin
Spasibo
Arigato

UniFAP