Modelos, Métodos e Técnicas de Engenharia de Software

Prof.: Sônia Aparecida Santana

una

Introdução

- Modularizar porções de código;
- Estabelecer ligações entre as partes;

Modularizar = Criar Módulos

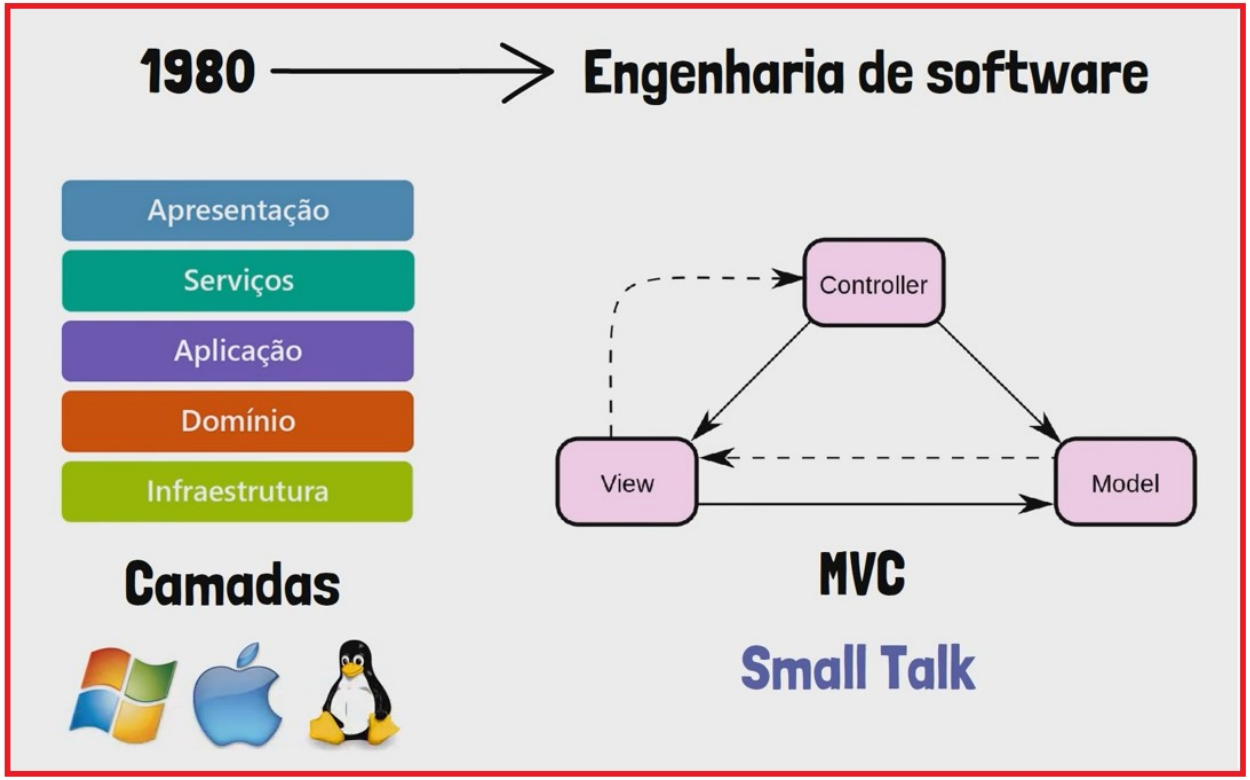
↓

Partes com atividades semelhantes

Atributos de Qualidade

- Desempenho
  - Localizar operações críticas e minimizar comunicações. Usar componentes de alta ao invés de baixa granularidade.
- Proteção
  - Usar uma arquitetura em camadas com itens críticos nas camadas mais internas.
- Segurança
  - Localizar características críticas de segurança em um pequeno número de subsistemas.
- Disponibilidade
  - Incluir componentes redundantes e mecanismos para tolerância à falhas.
- Facilidade de manutenção
  - Usar componentes substituíveis e de baixa granulariade.

Surgimento


1980
Engenharia de software
Apresentação
Serviços
Aplicação
Domínio
Infraestrutura
Camadas
Controller
View
Model
MVC
Small Talk

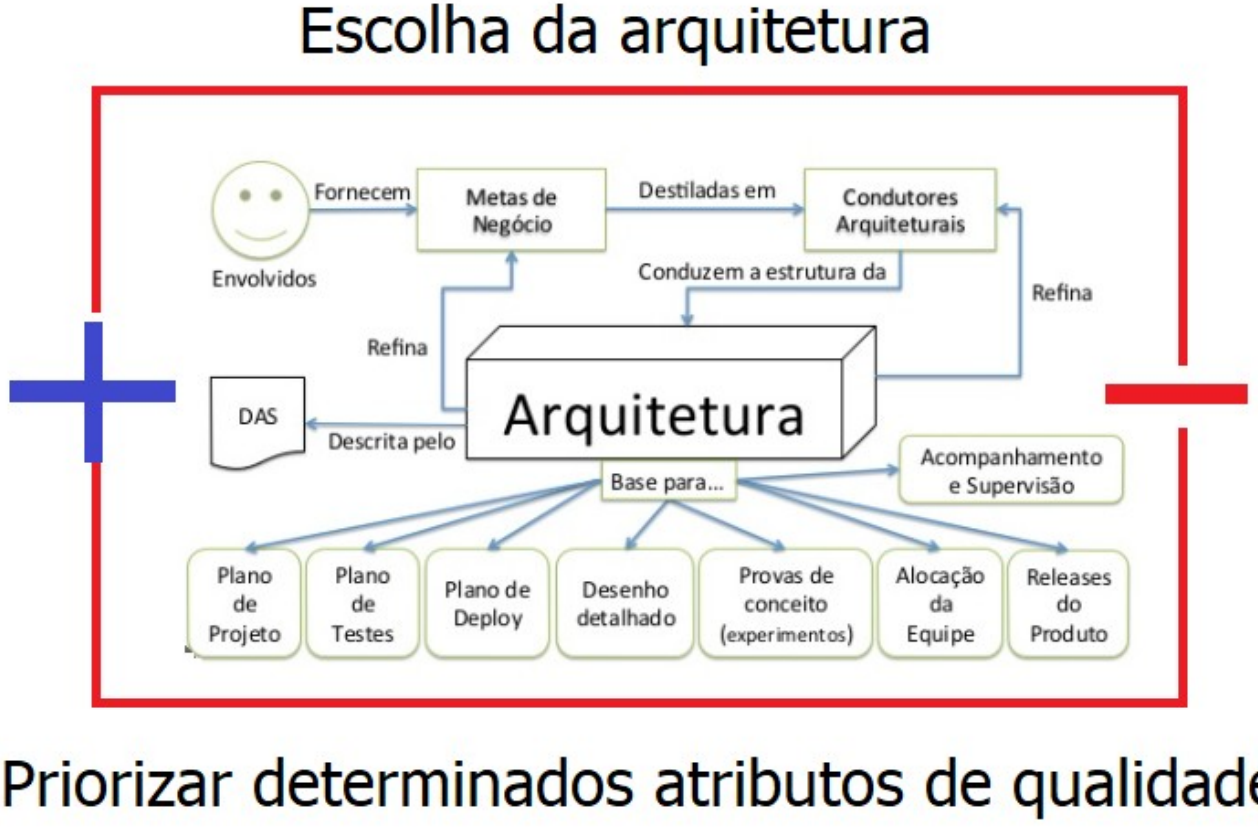
Arquitetura

- Projeto em mais alto nível
- Foco não são mais unidades pequenas (ex.: classes)
- Mas sim unidades maiores e mais relevantes
  - Pacotes, módulos, subsistemas, camadas, serviços, ...

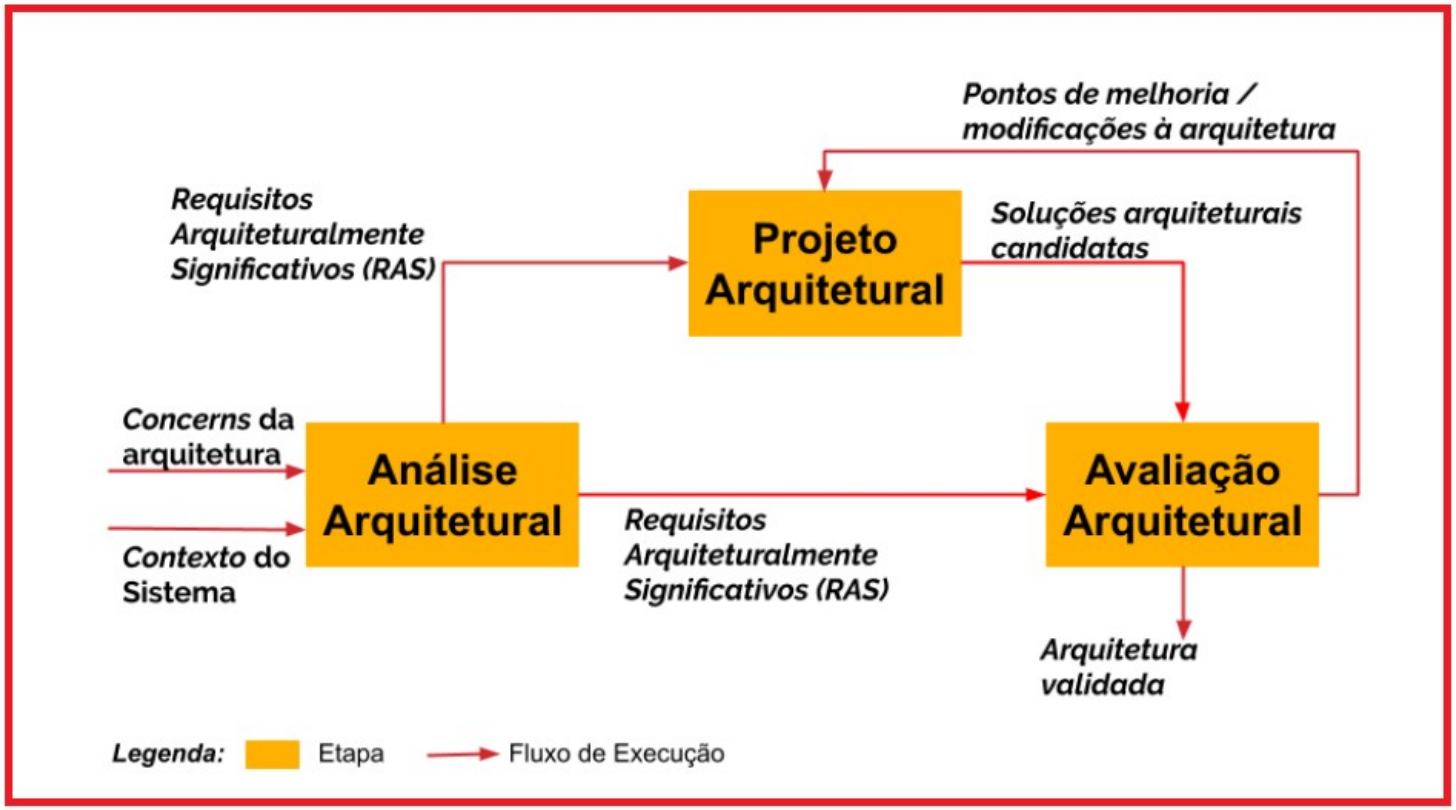
Relevância

- Definição de relevância depende do sistema
- Exemplo: banco de dados
  - Sistema de Informações: certamente é relevante
  - Sistema de Imagens Médicas: pode não ser relevante

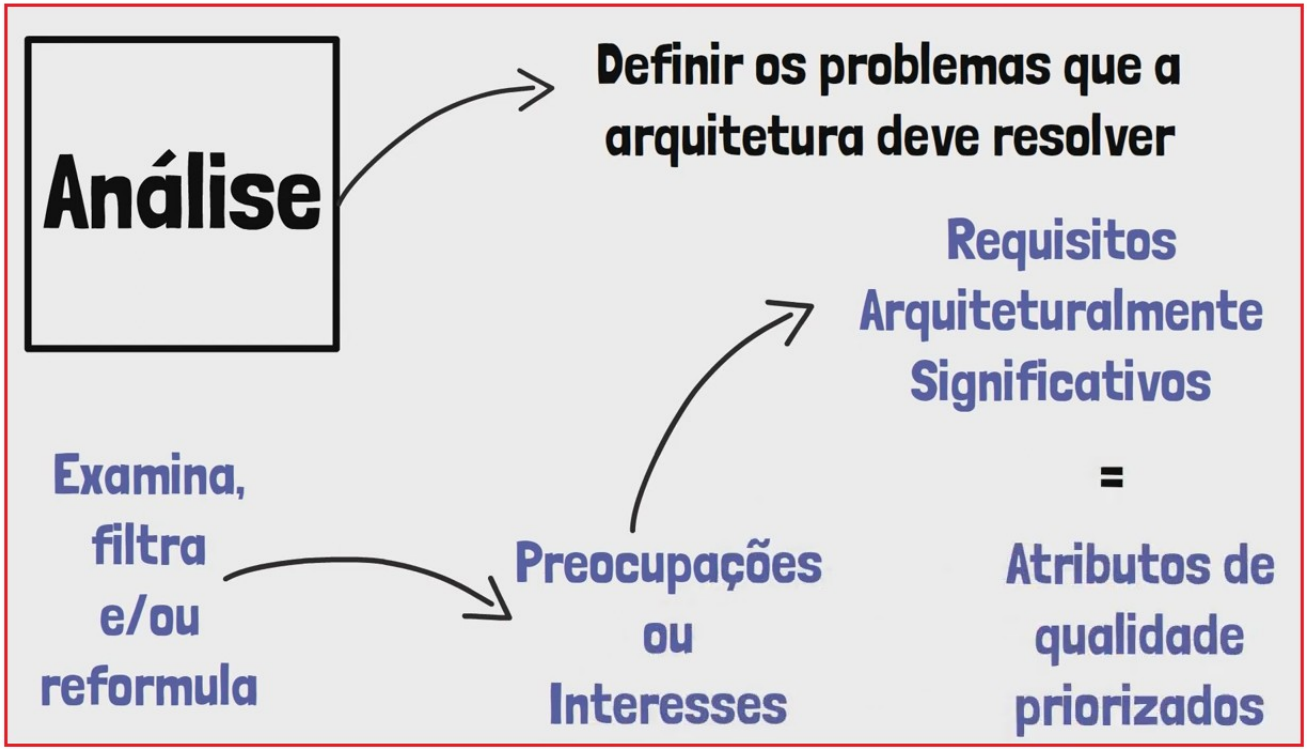
Introdução


Escolha da arquitetura
Fornecem
Metas de Negócio
Destiladas em
Condutores Arquiteturais
Envolvidos
Conduzem a estrutura da
Refina
Refina
DAS
Descrita pelo
Arquitetura
Base para...
Acompanhamento e Supervisão
Plano de Projeto
Plano de Testes
Plano de Deploy
Desenho detalhado
Provas de conceito (experimentos)
Alocação da Equipe
Releases do Produto
Priorizar determinados atributos de qualidade

Modelo de Processo Arquitetural



Etapa de Análise

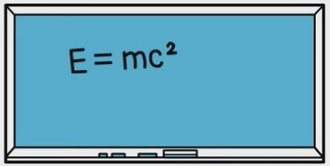


Etapa de Síntese

Síntese Arquitetural

Diferentes soluções ou alternativas arquiteturais são concebidas.

AQs Priorizados



Representando as principais partes e como elas são combinadas em diferentes arranjos para dar forma à arquitetura.

Etapa de Avaliação

Avaliação

Garante que as decisões de projeto feitas são as corretas e que, de fato, a arquitetura atende aos requisitos estabelecidos.

Arquitetura Candidata

Simulação

ATAM

Testes são realizados para mensurar critérios

Disponibilização da Arquitetura

O processo de concepção da arquitetura seja adaptado de acordo com o processo de desenvolvimento de software.

- Cascata
- Abordagens ágeis
- Iterativas/incrementais

Importante:

- Pensar nos AQs priorizados
- Uma arquitetura que atenda a eles
- Reutilizar

Architecture is about the important stuff.
Whatever that is. – Ralph Johnson

Importância de Arquitetura de Software

Debate Linus-Tanenbaum (1992)

Criador do sistema operacional Linux

Autor de livros e do sistema operacional Minix

Debate Linus-Tanenbaum (1992)

```
From: ast@cs.vu.nl (Andy Tanenbaum)
Newsgroups: comp.os.minix
Subject: LINUX is obsolete
Date: 29 Jan 92 12:12:50 GMT

I was in the U.S. for a couple of weeks, so I
LINUX (not that I would have said much had I
it is worth, I have a couple of comments now.
```

Argumento de Tanenbaum

- Linux possui uma **arquitetura monolítica**
- Quando o melhor seria uma **arquitetura microkernel**
- Monolítico: sistema operacional é um único arquivo
  - Gerência de processos, memória, arquivos, etc
- Microkernel: kernel só possui serviços essenciais
  - Demais serviços rodam como processos independentes

Resposta de Linus

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Subject: Re: LINUX is obsolete
Date: 29 Jan 92 23:14:26 GMT
Organization: University of Helsinki

Well, with a subject like this, I'm afraid I'll have to reply.
```

- Em teoria, arquitetura microkernel é mais interessante
- Mas existem outros critérios que devem ser considerados
- Um deles é que o Linux já era uma realidade e não apenas uma promessa

Nova Mensagem de Tanenbaum

- "Eu continuo com minha opinião. Projetar um kernel monolítico em 1991 é um erro fundamental."
- "Agradeça por não ser meu aluno. Se fosse, você não iria tirar uma nota alta com esse design."

Comentário Ken Thompson (Unix)

- "Na minha opinião, é mais fácil implementar um sistema operacional com um kernel monolítico."
- "Mas é também mais fácil que ele se transforme em uma bagunça à medida que o kernel é modificado."

Comentário de Linus

- Ken Thompson previu o futuro: 17 anos depois (2009) veja a declaração de Torvalds em uma conferência de Linux:
  - "Não somos mais o kernel simples, pequeno e hiper-eficiente que imaginei há 15 anos."
  - "Em vez disso, o kernel está grande e inchado. Quando adicionamos novas funcionalidades, o cenário piora."

Conclusão

- Moral da história: os "custos" de uma decisão arquitetural podem levar anos para aparecer...

Padrões Arquiteturais

- "Modelos" pré-definidos para arquiteturas de software
- Vamos estudar:
  - Camadas (duas e três camadas)
  - Model-View-Controller (MVC)
  - Microsserviços
  - Orientada a Mensagens
  - Publish/Subscribe

Arquitetura em Camadas

Arquitetura em Camadas

- Sistema é organizado em camadas, de forma hierárquica
- Camada *n* somente pode usar serviços da camada *n-1*
- Muito usada em redes computadores e sist. distribuídos

OSI model

Layer	Name	Example protocols
7	Application Layer	HTTP, FTP, DNS, SNMP, Telnet
6	Presentation Layer	SSL, TLS
5	Session Layer	NetBIOS, PPTP
4	Transport Layer	TCP, UDP
3	Network Layer	IP, ARP, ICMP, IPSec
2	Data Link Layer	PPP, ATM, Ethernet
1	Physical Layer	Ethernet, USB, Bluetooth, IEEE802.11

Vantagens

1. Facilita o entendimento, pois "quebra" a complexidade do sistema em uma estrutura hierárquica
2. Facilita a troca de uma camada por outra (ex.: TCP, UDP)
3. Facilita o reúso de uma camada (ex.: várias aplicações usam TCP).

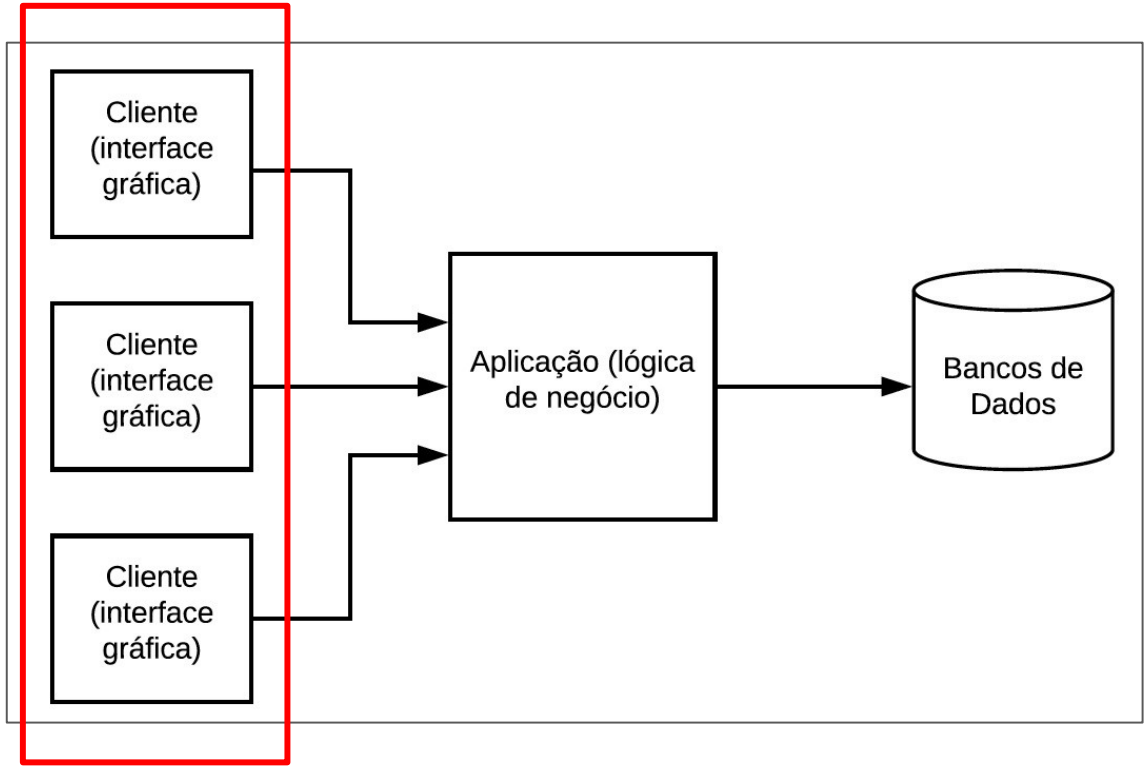
Variações para Sistemas de Informações

- Três camadas
- Duas camadas

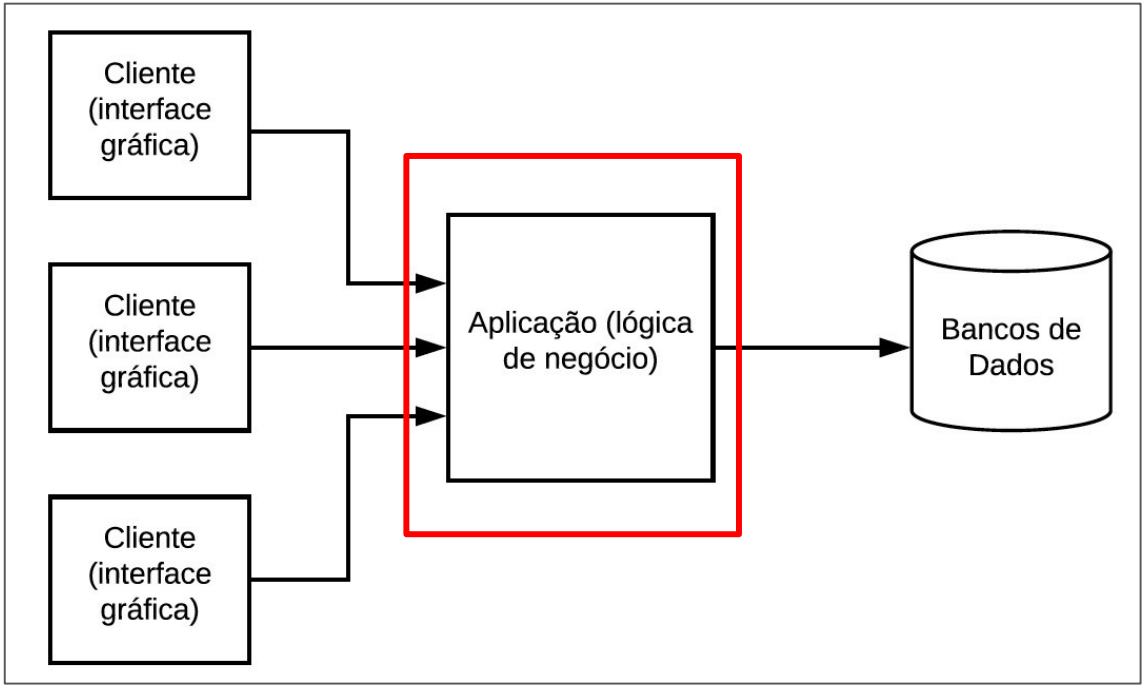
Arquitetura em Três Camadas

- Comum em processos de "downsizing" de aplicações corporativas nas décadas de 80 e 90
- Downsizing: migração de computadores de mainframes para servidores, rodando Unix

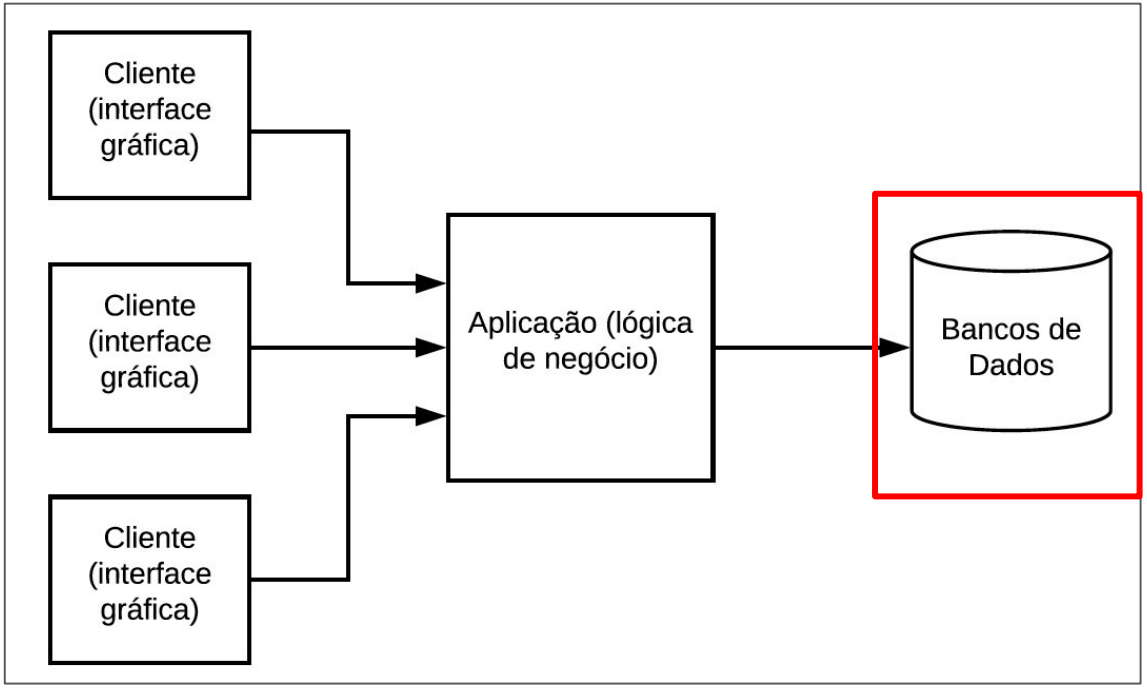
Arquitetura em 3 Camadas



Arquitetura em 3 Camadas



Arquitetura em 3 Camadas



Arquitetura em Duas Camadas

- Mais simples:
  - Camada 1 (cliente): interface + lógica
  - Camada 2 (servidor de BD): bancos de dados
- Desvantagem: todo o processamento é feito no cliente

Arquitetura Model-View-Controller (MVC)

Processo MVC

- O usuário interage com a interface gráfica (View). A interface gráfica interage com um intermediador (Controller), e este interage com o Model que executa as regras de negócios do sistema.

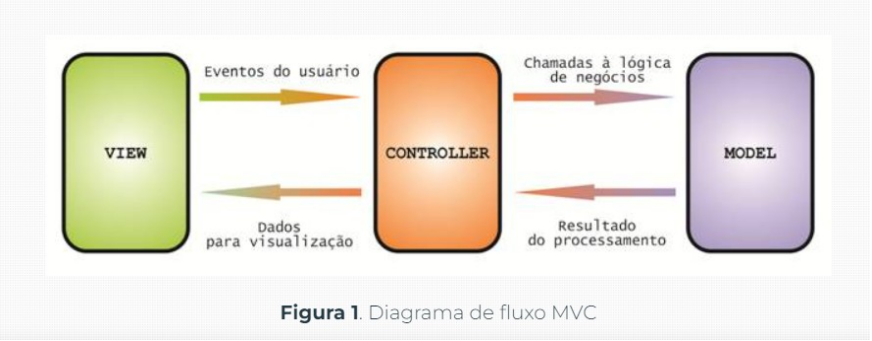


Figura 1. Diagrama de fluxo MVC

Por que Utilizar o Controlador?

- A ligação direta entre as duas camadas acarretaria para o código da interface duas responsabilidades, gerenciar a interface e também lidar com a lógica da camada Model.
- Isto aumentaria o acoplamento entre as duas camadas, deixando-as muito dependentes e assim, não seria possível a reutilização da interface com outro Model sem que fosse preciso modificar a camada de visualização.
- Quando se utiliza a camada Controller, a dependência entre o Model e a View são reduzidas ao máximo, possibilitando um projeto mais flexível, expansível e facilita futuras alterações.

Model

- O model é a camada que possui a lógica da aplicação.
- Ele é o responsável pelas regras de negócios, persistência com o banco de dados e as classes de entidades. O model recebe as requisições vindas do controller e gera respostas a partir destas requisições.

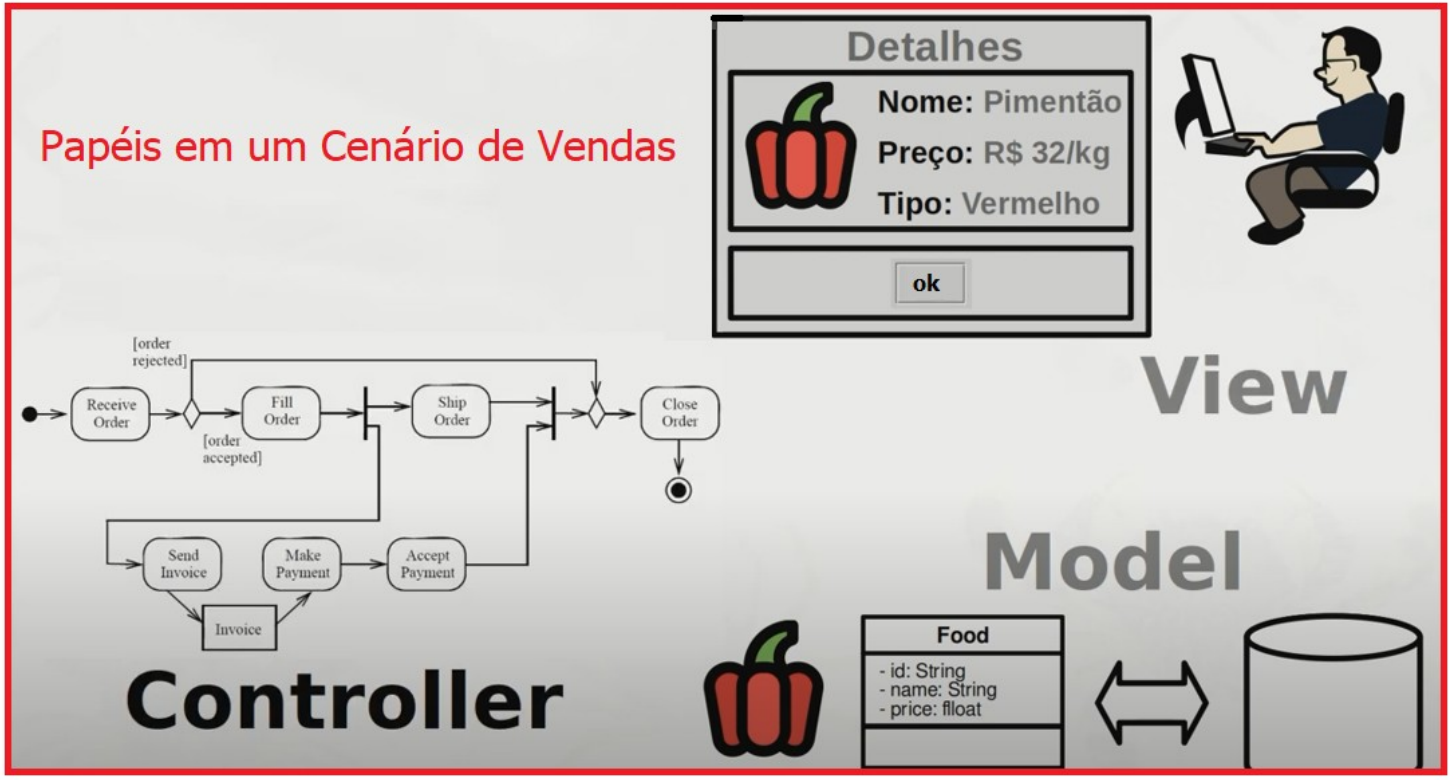
View

- A view é a camada de visualização e representa a parte do sistema que interage com o usuário.
- É pela interface que haverá a entrada dos dados inseridos pelo usuário e também a saída de informações que serão exibidas para ele. Esses dados serão inseridos ou exibidos geralmente por formulários de entrada ou de saída, tabelas, grids, entre outras formas. A view não contém lógica de negócios, portanto todo o processamento é feito pela camada model e então a resposta é repassada para a view pelo controlador.

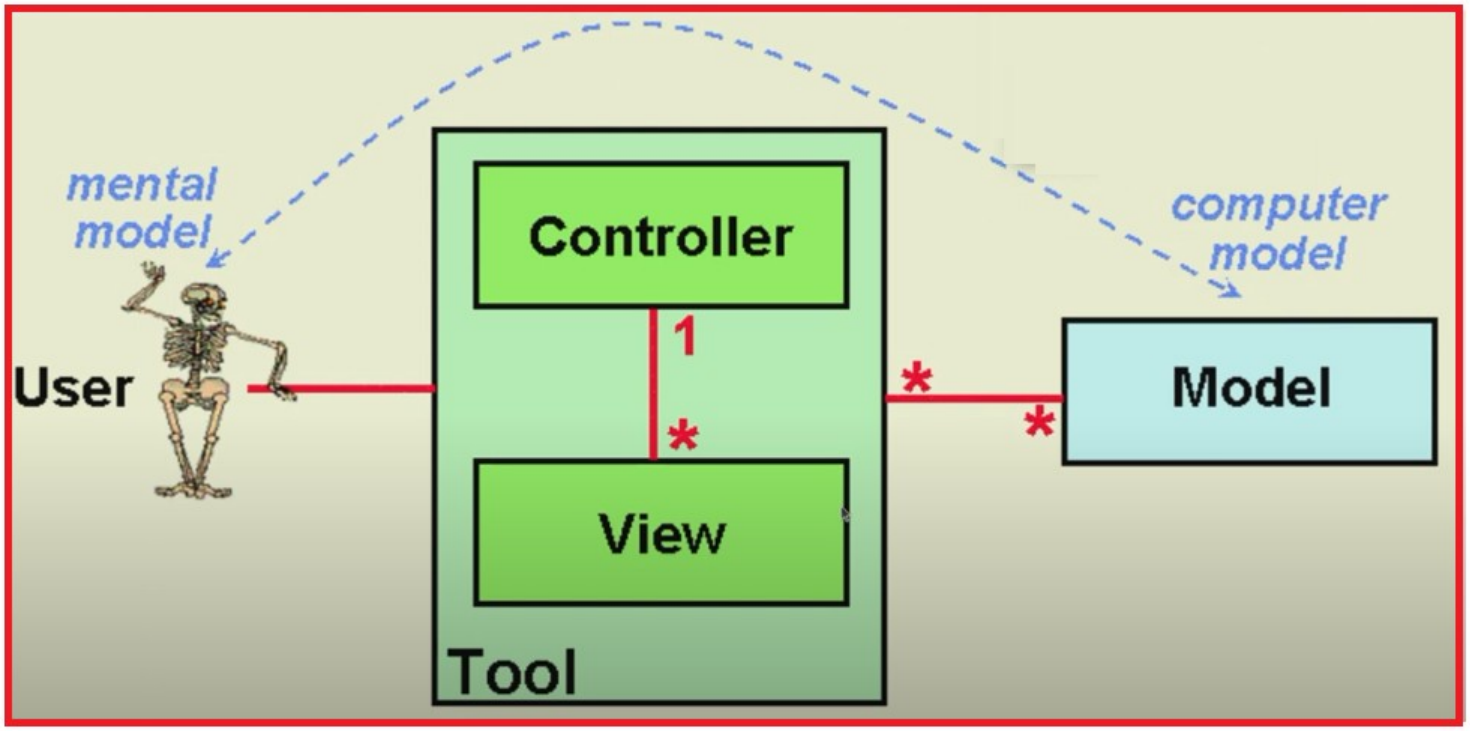
Controller

- Sua função é ser uma camada intermediária entre a camada de apresentação (View) e a camada de negócios (Model).
- Deste modo, toda requisição criada pelo usuário deve passar pelo controller, e este então se comunica com o model. Se o model gerar uma resposta para essas requisições, ele enviará as respostas ao controller que por sua vez repassa à camada view.

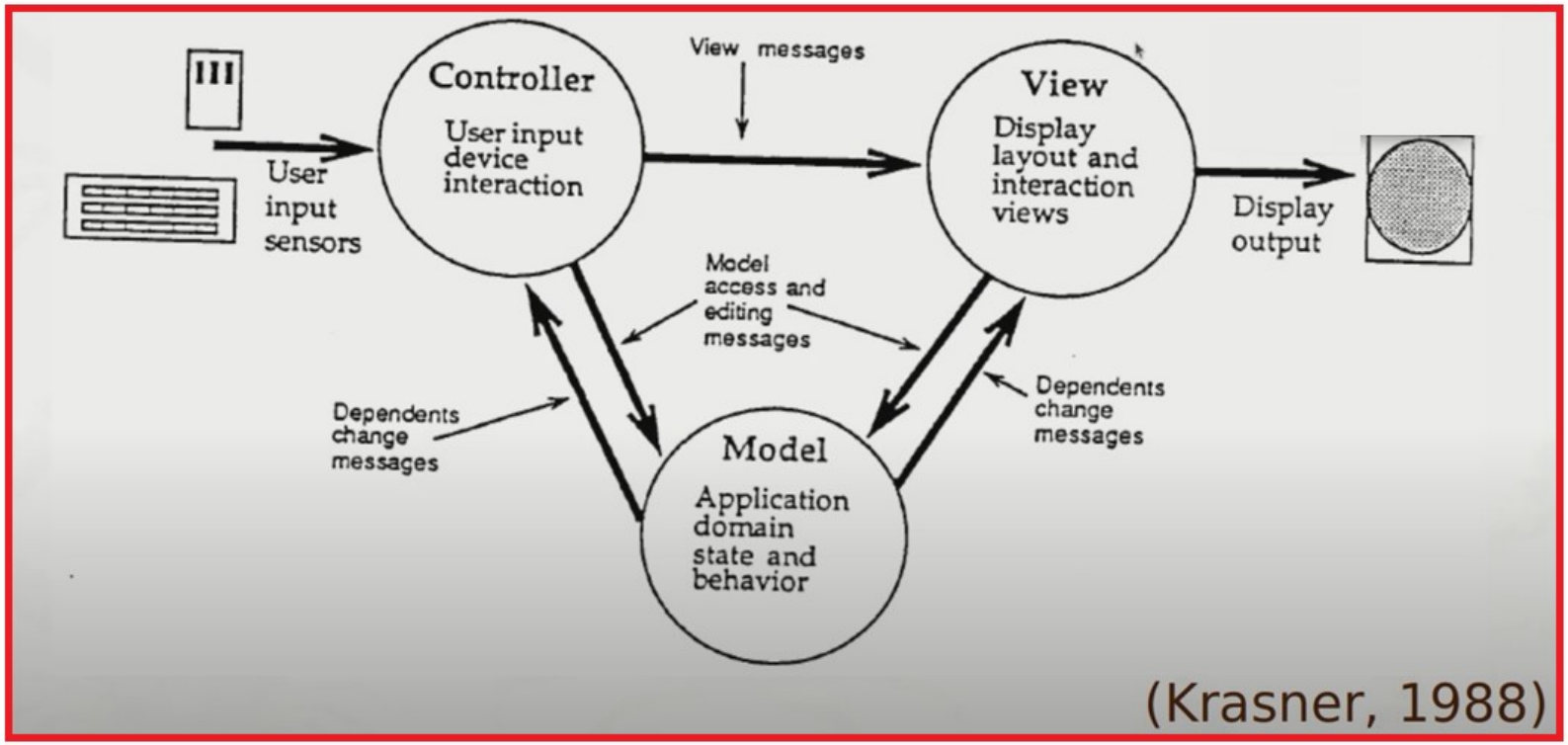
MVC - Exemplo


Papéis em um Cenário de Vendas
Detalhes
Nome: Pimentão
Preço: R$ 32/kg
Tipo: Vermelho
ok
View
[order rejected]
Receive Order
Fill Order
[order accepted]
Ship Order
Close Order
Send Invoice
Make Payment
Accept Payment
Invoice
Controller
Model
Food
- id: String
- name: String
- price: float

MVC - Interpretações



MVC: Krasner


Controller
User input device interaction
View messages
View
Display layout and interaction views
User input sensors
Display output
Model access and editing messages
Dependents change messages
Dependents change messages
Model
Application domain state and behavior
(Krasner, 1988)

Exercícios de Fixação

- Qual é o aspecto mais importante da arquitetura MVC?
- Como funciona a arquitetura MVC-Web?
- Qual a diferença entre o estilo arquitetural MVC e 3 camadas?

Exercícios de Fixação

- Seja o seguinte cenário:
  - Clientes compram mercadorias em uma loja;
  - A mercadoria é vendida pelos funcionários da loja;
  - Os funcionários da loja ganham comissão sobre suas vendas;
  - O gerente da loja compra mercadorias de seus fornecedores;
  - O gerente da loja controla o estoque das mercadorias;

Exercício de Fixação

- Citar exemplos de possíveis classes de interface, controler e model.