Aula 9 - Padrões Arquiteturais

SSC0620/SSC5764 - Engenharia de Software
1º Semestre de 2019
Profa. Dra. Elisa Yumi Nakagawa e Prof. Dra. Lina Garcés
PAE: Brauner Oliveira e Daniel Santos

12 de Abril de 2019

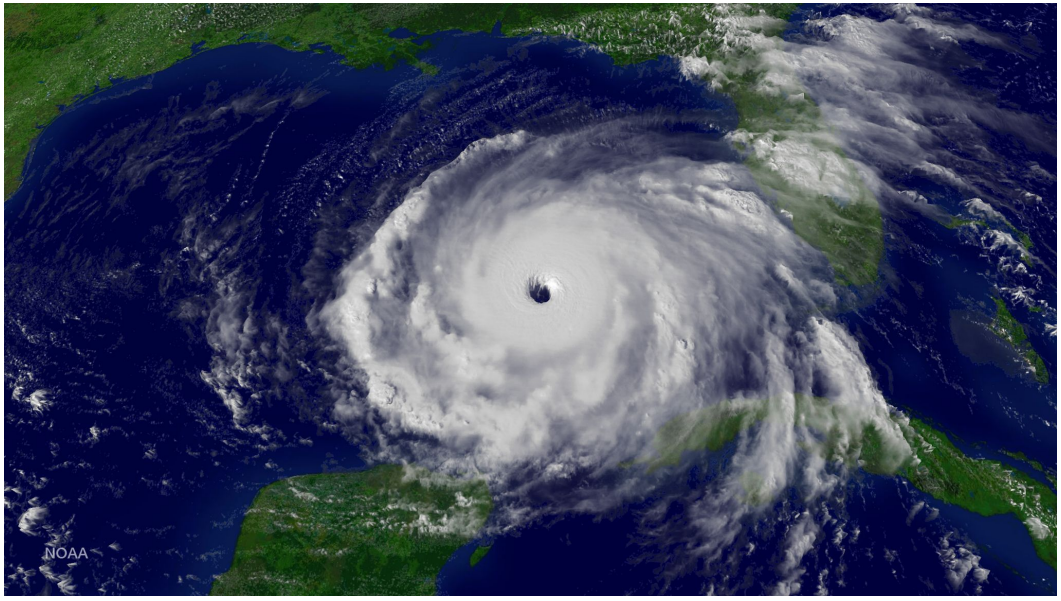
O que é um padrão?

Segundo o dicionário da Cambridge,
um padrão é:

uma forma particular em que alguma coisa/situação:

- **é realizada, ou**
- **é organizada, ou**
- **acontece**
  de forma repetitiva e semelhante

Padrões na natureza



Padrões em Sistemas Software

Sistemas de Mensagens Instantâneas

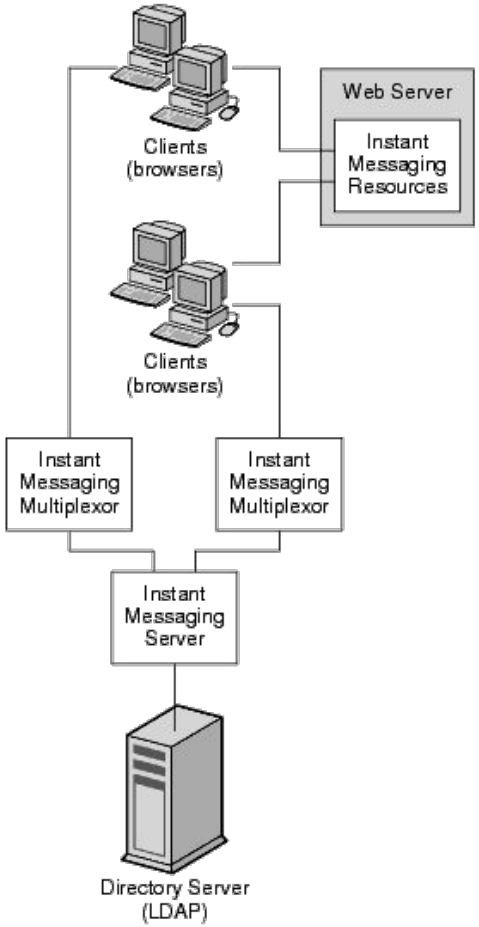
Características comuns:

Disponibilidade → Envio instantâneo das mensagens
Sistemas **distribuídos**
Flexibilidade → Multiplataforma
Escalabilidade → Milhões de usuários
Gratuito

Padrões em Sistemas Software

Sistemas de Mensagens Instantâneas

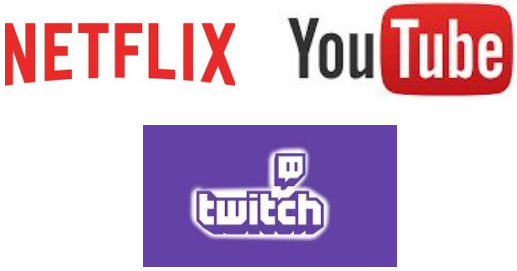
Exemplo de arquitetura



https://docs.oracle.com/cd/E19142-01/819-0063/im-architecture.html

Padrões em Sistemas Software

Sistemas de Streaming de Vídeo

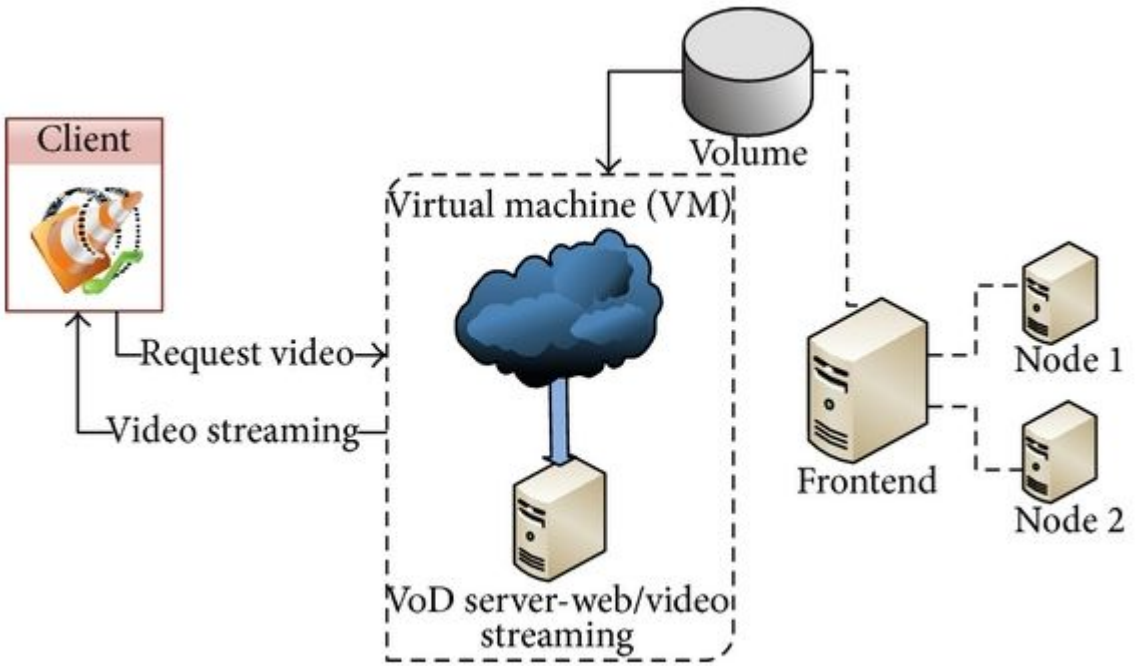


Características comuns:

Desempenho → Envío em “tempo real” dos frames de vídeo, baixa latência, evitar perda de frames
Modularidade → Sistemas distribuídos
Flexibilidade → Sistemas Multiplataforma (Apps, Web)
Escalabilidade → Milhões de usuários, variedade de vídeos de boa qualidade (alto armazenamento)
Disponibilidade → Serviço sempre on-line

Padrões em Sistemas Software

Sistemas de Streaming de Vídeo



Melo, Rosangela & Bezerra, Maria Clara & Dantas, Jamilson & Matos, Rubens & Melo Filho, Ivanildo & Maciel, Paulo. (2014). Redundant VoD Streaming Service in a Private Cloud: Availability Modeling and Sensitivity Analysis. Mathematical Problems in Engineering. 2014. 10.1155/2014/764010.

O que é um padrão arquitetural?

O que é um padrão arquitetural?

Descreve uma **solução arquitetural** para **problemas recorrentes** em sistemas software que se apresentam em **contextos** específicos.

A solução arquitetural do padrão define:

- as estruturas de software (componentes, serviços, módulos)
- responsabilidades e relações entre essas estruturas
- formas que as estruturas colaboram para solucionar o problema

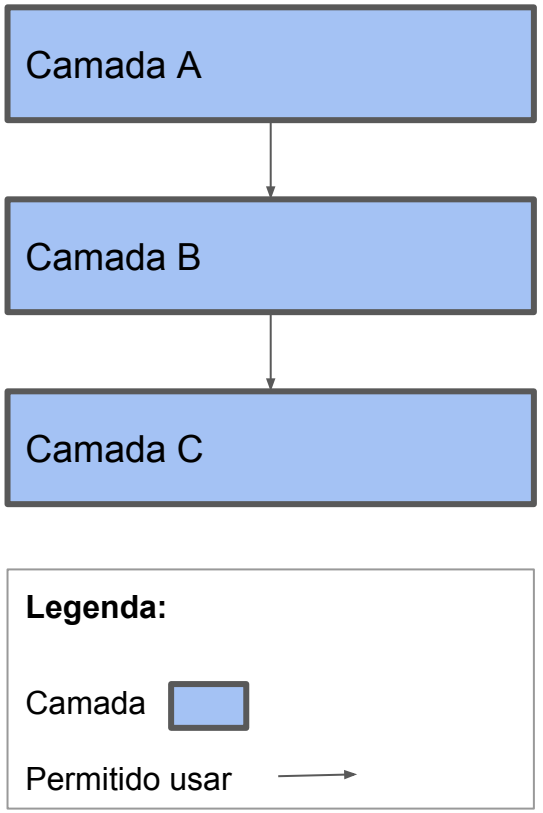
O que é um padrão arquitetural?

Um padrão arquitetural é um **conjunto de decisões** de projeto de software que define:

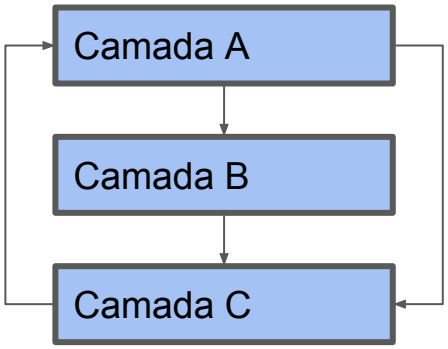
- **Utilidade**
- tipos de **elementos** (componentes, serviços, módulos),
- tipos de **relações** (dependência, colaboração, controle),
- **propriedades** dos elementos (responsabilidades, funcionalidades), e
- **restrições** sobre como os elementos se relacionam (comunicação unidirecional, bidirecional)

Exemplo do padrão ***Layers*** (camadas):

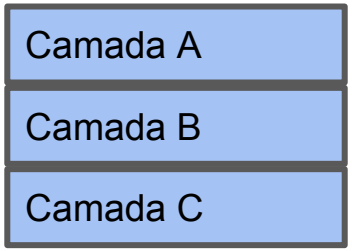
Descrição	**Agrupar** entidades de software (módulos, componentes) em camadas de funcionalidades e permitir seu uso entre elas.
Utilidade	Promover modificações, portabilidade, reuso, separação de interesses, gerenciamento de complexidade, facilidade de comunicação e entendimento das estruturas.
Elementos	Camadas → cada camada deve explicar quais entidades de software ela contém.
Relações	Permitido usar → elementos em uma camada são permitidos de usar elementos de outra camada
Propriedades	Camadas generalizam características das entidades de software alocadas nelas.
Restrições	Cada elemento de software é alocado em **UMA** camada. Para usar este padrão devem ser definidas **pelo menos DUAS c**amadas. A relação *"permitido usar"* **NÃO** deve ser circular, por exemplo elementos da camada C não podem usar elementos da camada A



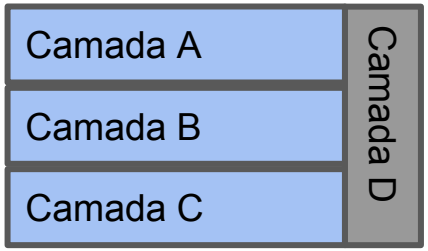
Esses exemplos seguem o padrão ***Layers*** ?



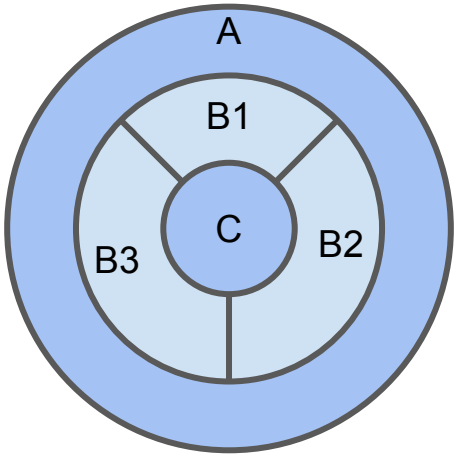
Exemplo 1



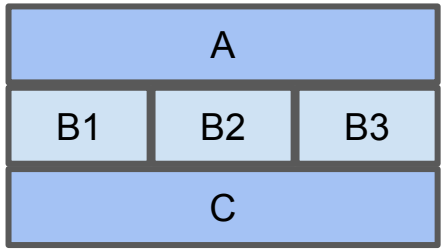
Exemplo 2



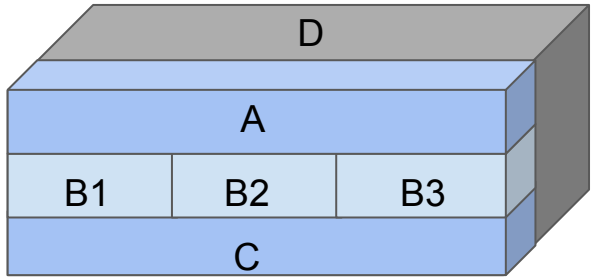
Exemplo 3



Exemplo 4

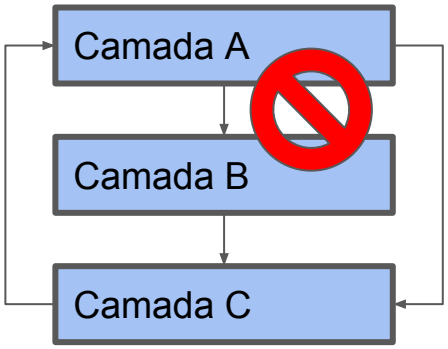


Exemplo 5

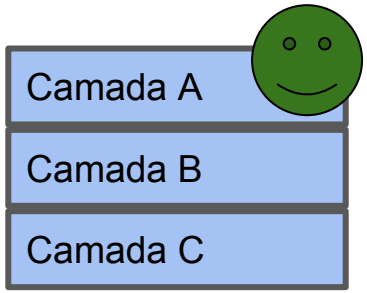


Exemplo 6

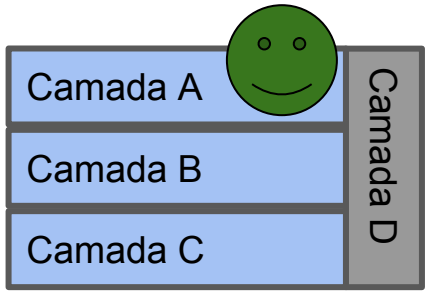
Esses exemplos seguem o padrão ***Layers*** ?



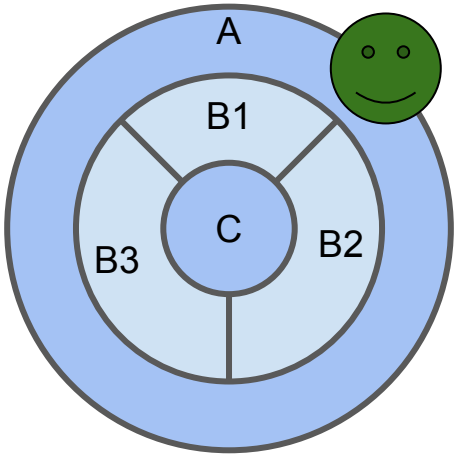
Exemplo 1



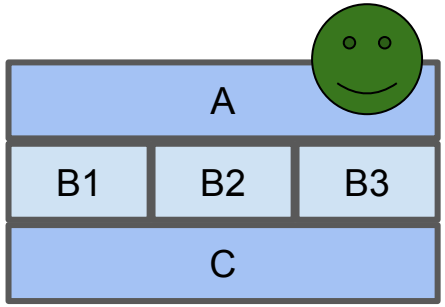
Exemplo 2



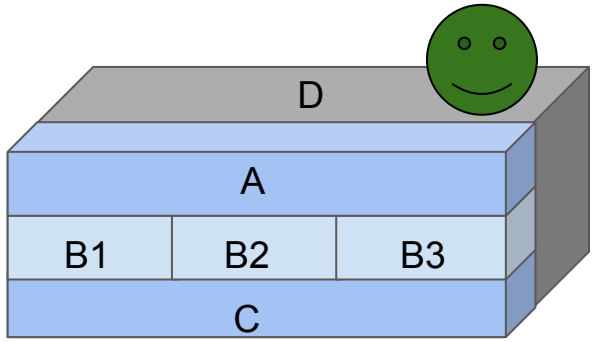
Exemplo 3



Exemplo 4

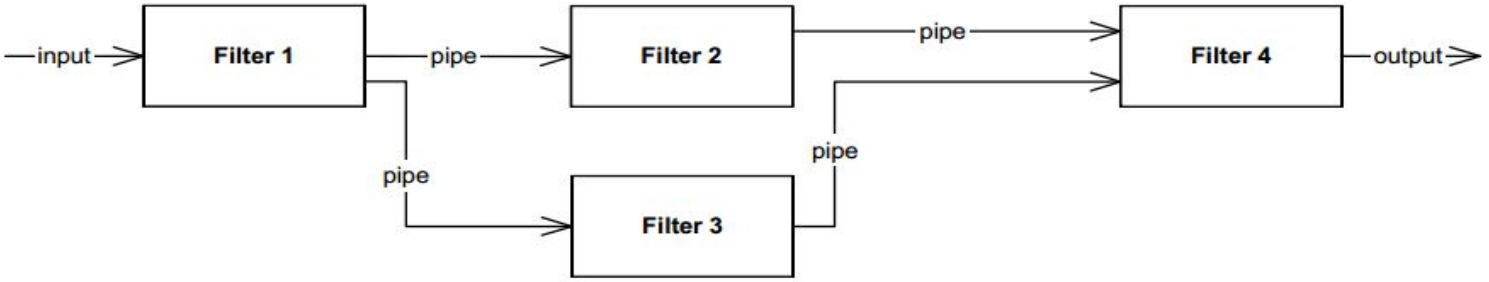


Exemplo 5



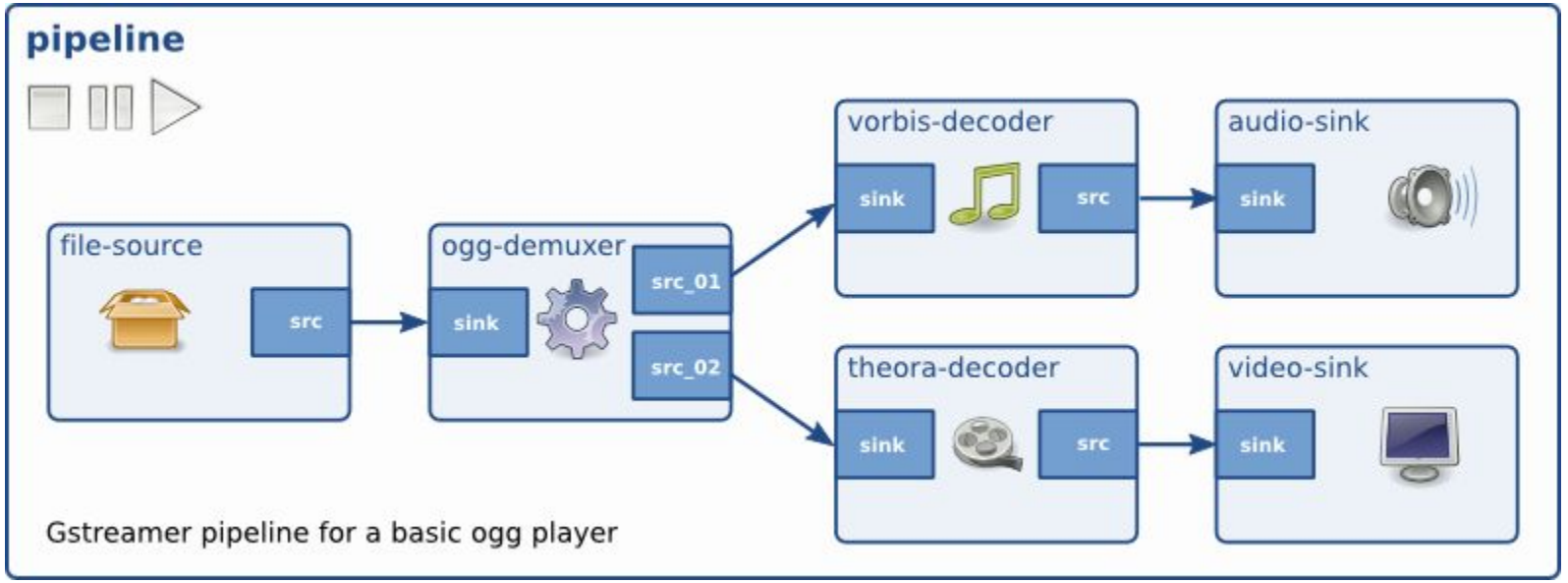
Exemplo 6

Exemplo do padrão *Pipe-Filter* (PF):

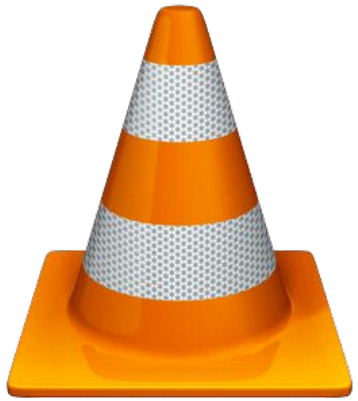


Descrição	Possibilita a transformação de fluxo de dados de forma consecutiva
Utilidade	Melhora o reuso da implementação dos filtros devido a baixa dependência entre eles; favorece a paralelização do processamento dos dados; simplifica o raciocínio sobre o comportamento geral do sistema
Elementos	Filtros → componente que transforma dados de entrada e comunica as transformações como um novo conjunto de dados de saída. Pipe → Conecta a saída de um filtro com a entrada de outro. Responsável pela comunicação do streaming de dados.
Relações	Relação de *attachment* (ligação) que linka os pontos de saída dos filtros com os pontos de entrada do pipe, e os pontos de entrada dos filtros com os pontos de saída do pipe
Propriedades	Dados são transformados pelos filtros e comunicados pelas pipes.
Restrições	Os dados de saída do filtro emissor devem ser compatíveis com os dados de entrada do filtro receptor.

Exemplo do padrão ***Pipes-and-Filters*** (PF):

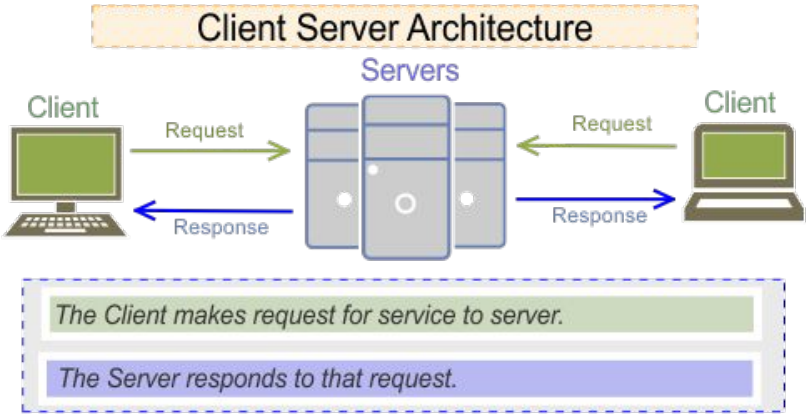


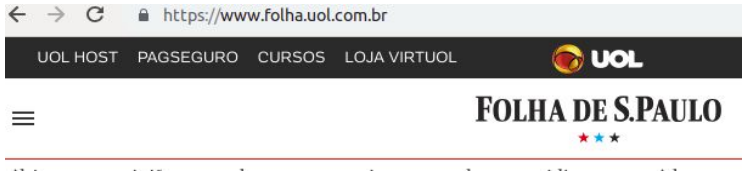
Reprodutores de vídeo em diferentes formatos



Exemplo do padrão *Cliente-Servidor* (CS):

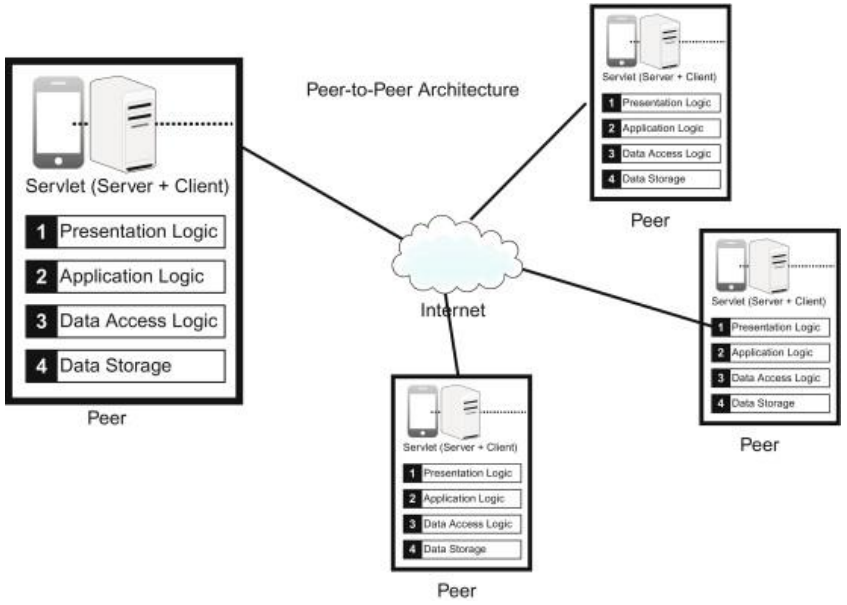
Descrição	Permite que dois componentes interajam através de solicitações Request/Reply
Utilidade	Facilita modificações e reuso dos componentes, melhora a escalabilidade, disponibilidade (réplicas de servidores), segurança
Elementos	Clientes → invoca serviços Servidores → fornece serviços Request/Reply Conector → usado para invocar serviços e enviar respostas
Relações	Relação de *attachment* (ligação) que linka os portos *request* do cliente - conector com os portos *reply* do conector - servidor
Propriedades	O cliente inicia as interações, invocando serviços dos servidores e esperando as respostas
Restrições	Clientes conectados através de conectores tipo Request/Reply; Servidores atendem solicitações e fornecem respostas aos clientes Componentes podem ser alocados em camadas.





Exemplo do padrão *Peer-to-Peer* (P2P):

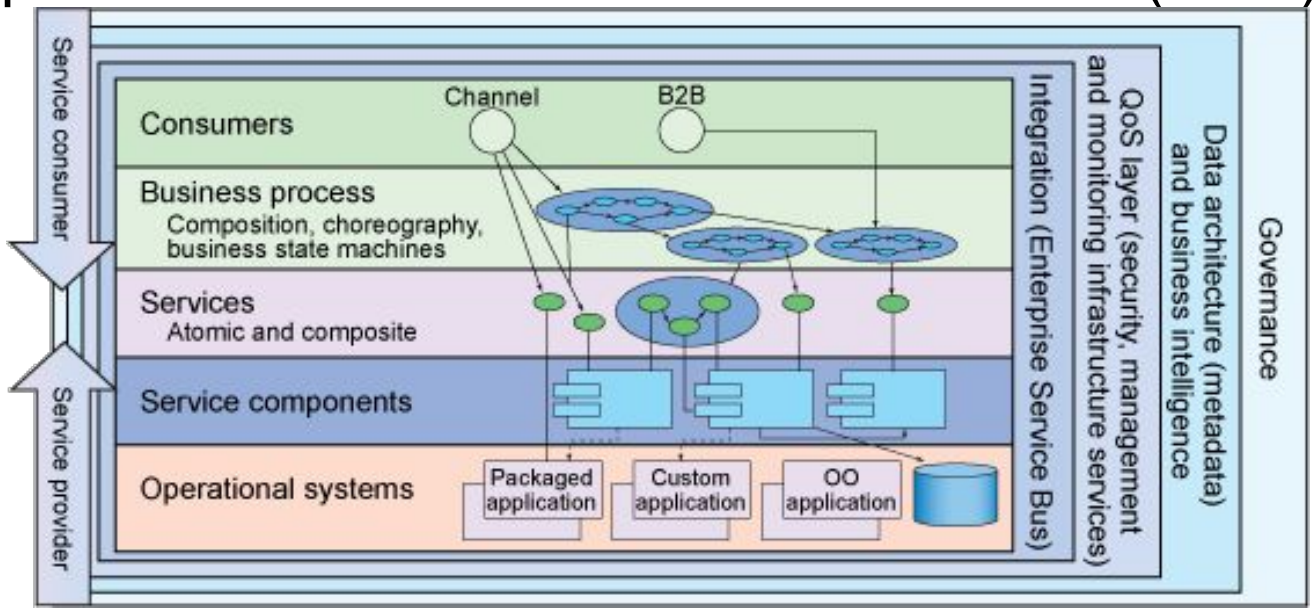
Descrição	Componentes interagem diretamente com pares (peers), intercambiando serviços
Utilidade	Fornecer alta disponibilidade, escalabilidade. Possibilita a interação entre sistemas totalmente distribuídos (ex. grid computing)
Elementos	Componente ***Peer*** Conector ***Call-Return*** usado para conectar o peer à rede, procurar outros peers, e invocar serviços de outros peers.
Relações	Peers invocam e oferecem serviços através dos conectores call-return
Propriedades	Links podem mudar em tempo de execução devido a mudanças no desempenho
Restrições	Cada peer deve implementar toda a lógica de servidor. É necessário que um peer tenha o perfil de roteador para conectar novos peers.



Grid computing caseiras

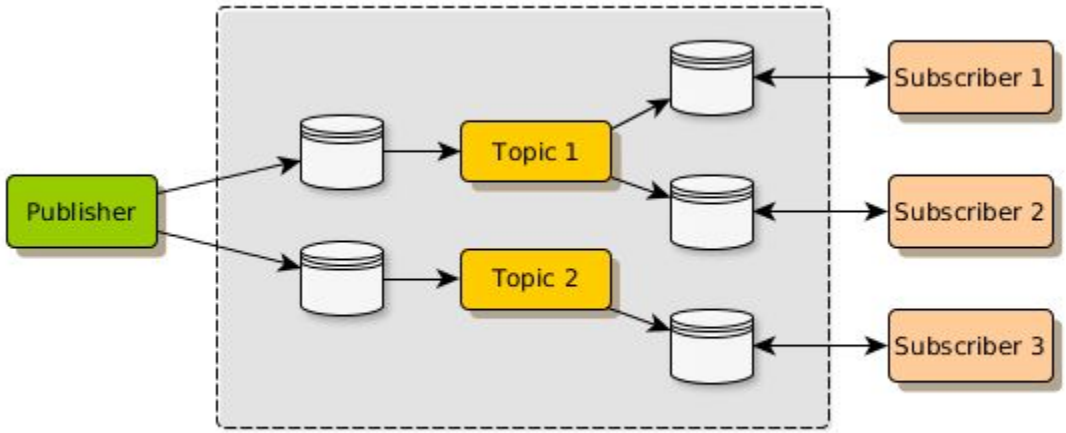
https://www.sciencedirect.com/topics/computer-science/peer-to-peer-architectures

Exemplo do padrão *Service-Oriented Architecture* (SOA):



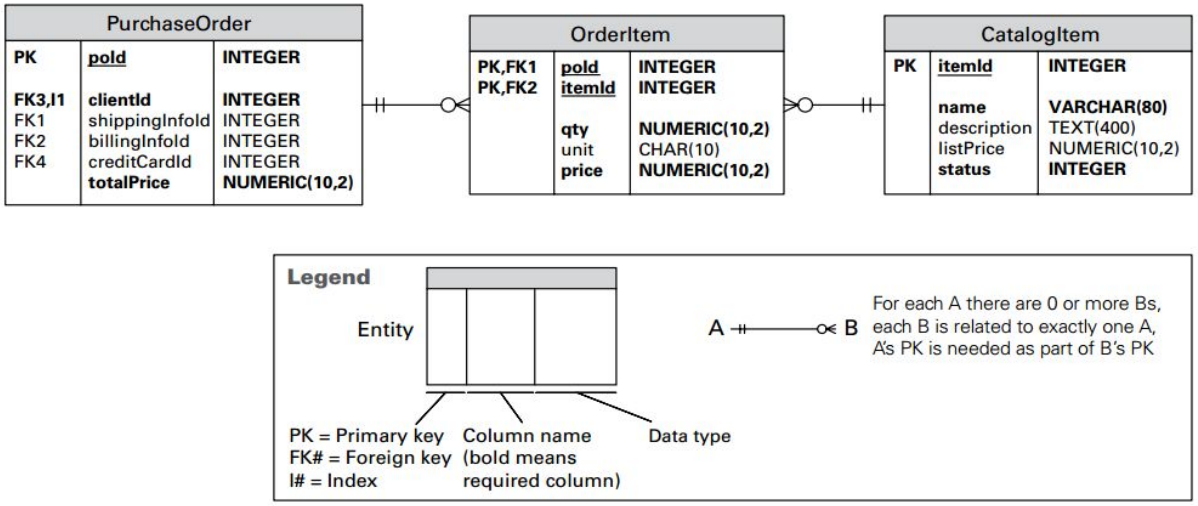
Descrição	Coleção de componentes distribuídos que fornecem/consomem serviços
Utilidade	Favorece a interoperabilidade de componentes distribuídos executados em diferentes plataformas na internet
Elementos	fornecedores de serviços, consumidores de serviços, ESB (enterprise service bus / barramento), registro de serviços, servidor de orquestração, conectores (SOAP, REST), conectores de mensagem
Relações	Serviços consomem e fornecem funcionalidades através de solicitações enviadas usando conectores SOAP, REST, RESTFUL, etc.
Propriedades	Serviços cooperam para executar processos de negócio e criar novos serviços compostos coordenados pelo orquestrador.
Restrições	Serviços são conectados somente através de intermediários (ESB) que servem como hub.

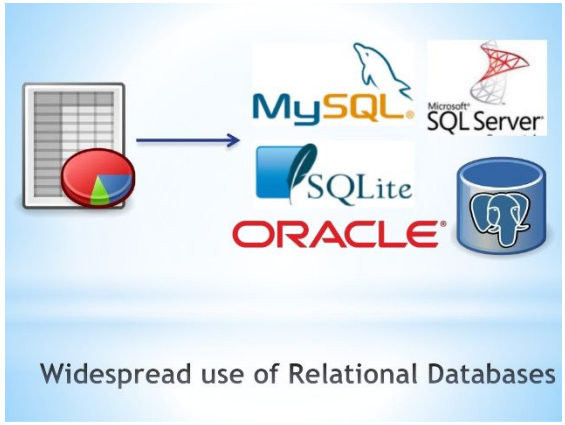
Exemplo do padrão ***Publish-Subscribe*** (P-S):



Descrição	Componentes interagem através de eventos de interesse (tópicos) aos quais foram subscritos
Utilidade	Enviar eventos diretamente a componentes interessados sem precisar estabelecer uma comunicação direta.
Elementos	Subscribers → componente interessado em receber eventos Publishers → Componentes que comunicam os eventos Conectores P-S → possui perfil de anunciante e de ouvinte de eventos Tópico → Tipo de evento que será comunicado
Relações	Relação de *attachment* (ligação) que associa componentes com o conector P-S, descrevendo quais componentes estão interessados em receber e publicar eventos relacionados a tópicos específicos.
Propriedades	Subscribers se inscrevem em tópicos. Quando um evento relacionado a um tópico é anunciado pelo publisher, o conector P-S despacha o evento a todos os subscribers
Restrições	Publishers e subscribers se conectam através dos conectores P-S a um barramento responsável por identificar e encaminhar os tópicos.

Exemplo do padrão ***Data Model*** (ER):

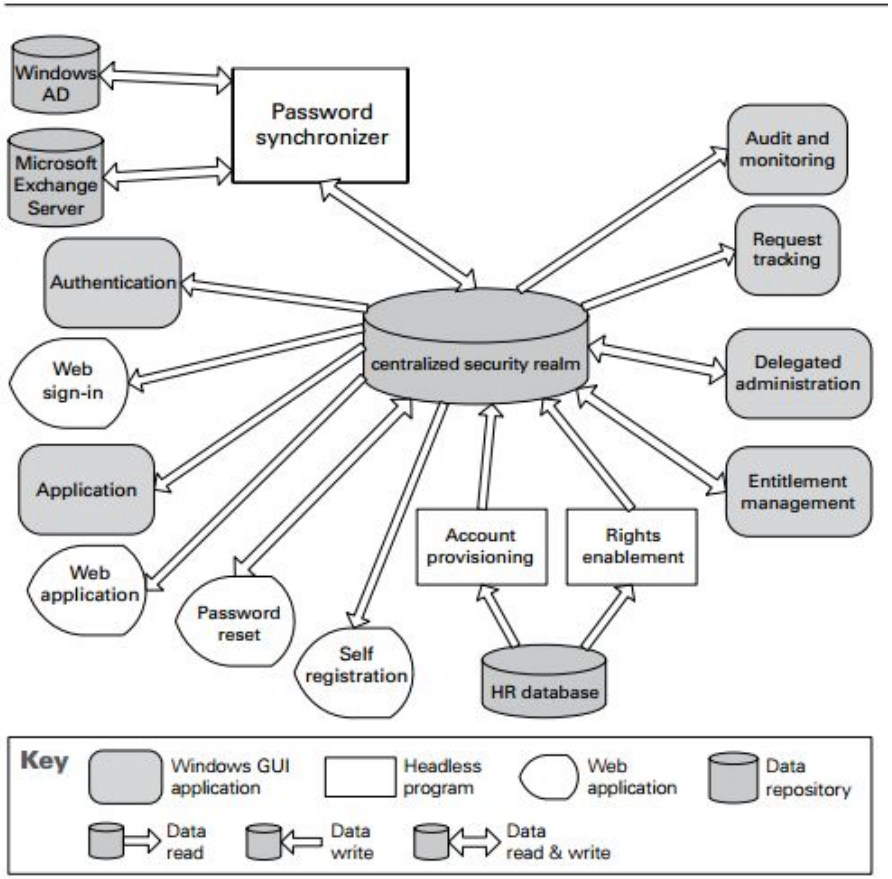


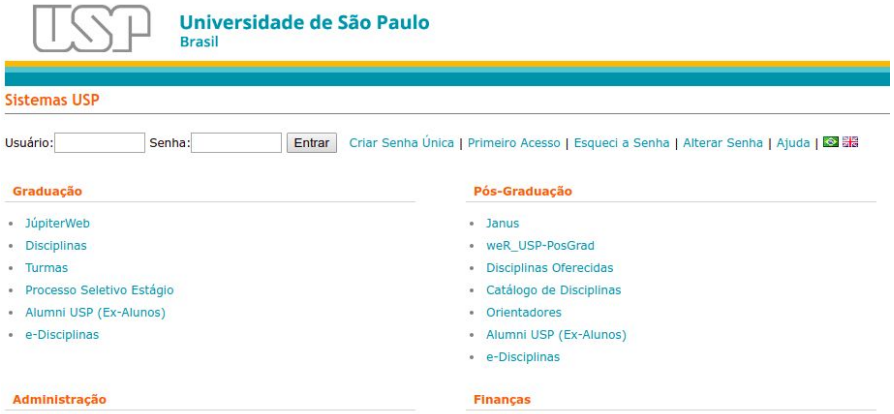


Descrição	Esse padrão descreve as estruturas de dados e suas relações
Utilidade	Comunicação das estruturas de dados a serem usadas na base de dados / repositório; ajuda analisar impactos das mudanças nas estruturas (tabelas, atributos, tipos); ajuda evitar redundância e inconsistência de dados; guia para a implementação dos módulos/componentes que acessam os dados.
Elementos	Entidades de dados e suas propriedades (nome, atributos, PK, regras de acesso)
Relações	One-to-one, one-to-many, and many-to-many; Generalização; Agregação
Propriedades	Uma entidade de dado representa informação necessária para o sistema funcionar
Restrições	Evitar modificações tardias

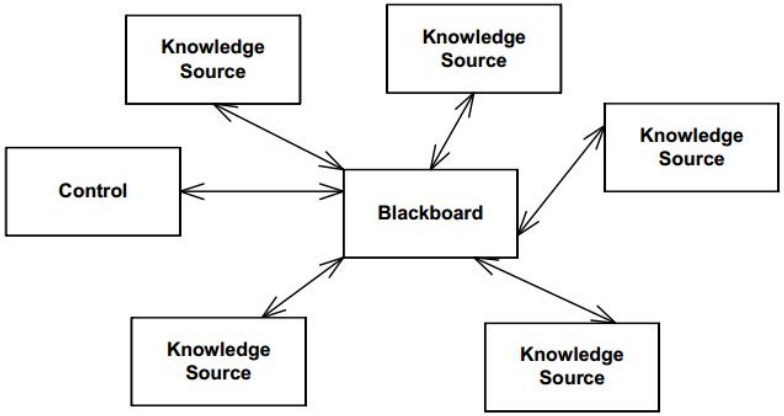
Exemplo do padrão ***Shared-Data*** (SD):

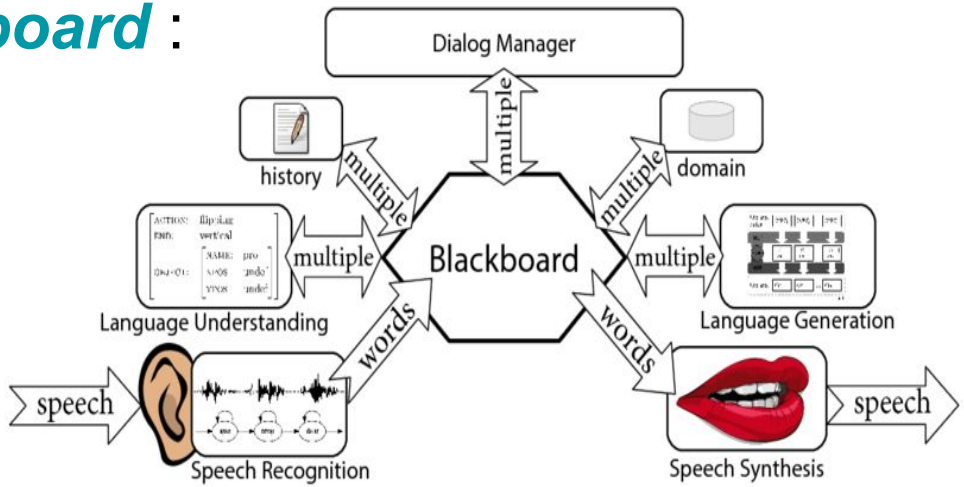
Descrição	Componentes interagem através de um repositório de dados compartilhados
Utilidade	Possibilita múltiplos componentes acessar a dados persistentes, e favorece a modificação de produtores e consumidores de dados
Elementos	Componente repositório, componente que acessa os dados, conector que escreve e lê os dados.
Relações	Relação de *attachment* (ligação) que determina quem pode acessar o repositório
Propriedades	Dados são persistentes no repositório. O controle de acesso é realizado pelo repositório.
Restrições	Componentes não interagem diretamente entre eles, somente através do repositório





Exemplo do padrão ***Blackboard*** :





Descrição	É um tipo de repositório compartilhado que utiliza as informações dos seus clientes para gerar novo conhecimento o qual pode ser acessado de novo por seus clientes para refinar suas operações.
Utilidade	Compartilhamento de informação gerada durante a execução de tarefas complexas com diversos clientes. Ex. reconhecimento de voz, reconhecimento de imagens ...
Elementos	Blackboard → Divide tarefas complexas em tarefas determinísticas e gera uma solução para ser compartilhada com os clientes Cliente→ gera informações e envia ao blackboard. Também consulta as informações/conhecimento gerado pelo blackboard Componente de controle → monitora o blackboard e coordena os clientes de acordo ao estado do blackboard
Relações	Clientes, componente de controle, e o blackboard interagem através de conectores de escritura e leitura
Propriedades	O blackboard usa os resultados de seus clientes para executar heurísticas e gerar conhecimento.
Restrições	Clientes somente podem acessar aos dados resultantes da execução das tarefas do blackboard. Clientes não podem acessar a dados enviados por outros clientes. Não há interação direta entre clientes.

Um padrão é suficiente para sua arquitetura?

Um padrão é suficiente para sua arquitetura?

R: Depende dos **requisitos de qualidade** importantes na arquitetura.

Na maioria das vezes é necessário **combinar** vários padrões para atingir as qualidades desejadas.

Como selecionar padrões para formar a arquitetura do seu sistema?

Como selecionar padrões para formar a arquitetura do seu sistema?

R: Cada padrão **soluciona** um problema específico, **favorecendo** um conjunto de **qualidades**.

Dependendo dos requisitos de qualidade da arquitetura, é necessário analisar quais padrões são os mais favoráveis.

Veja o seguinte exemplo para Big Data ...

Requisitos de **qualidade** vs. **Padrões** arquiteturais no contexto de Big Data

Table II. Quality Requirements x Architectural Patterns

Characteristics of Big Data Systems	Quality Attributes	Layered	Shared Repository	Pipe and Filters	Broker
,*Volume, Velocity*	*Scalability*	–	–	–	+
Volume, Velocity	*Performance*	–	–	–	–
Volume	*Modularity*	+	–	+	
Veracity	*Consistency*		+	–	
Veracity	*Security*	+	+	–	–
Velocity	*Real-time*	–	–	–	
Variety, Value	*Interoperability*				+
Value	*Availability*		–		

Bruno Sena, Lina Garcés, Ana Allian, Elisa Y. Nakagawa. Investigating the applicability of architectural patterns in big data systems. In: Proceedings of the 25th Conference on pattern language of programs (PLoP), 2018.

Requisitos de **qualidade** vs. **Padrões** arquiteturais no contexto de Big Data

Conflitos / "perde-ganha" associados à escolha de um padrão

Table II. Quality Requirements x Architectural Patterns

Characteristics of Big Data Systems	Quality Attributes	Layered	Shared Repository	Pipe and Filters	Broker
,*Volume, Velocity*	*Scalability*	–	–	–	+
Volume, Velocity	*Performance*	–	–	–	–
Volume	*Modularity*	+	–	+	
Veracity	*Consistency*		+	–	
Veracity	*Security*	+	+	–	–
Velocity	*Real-time*	–	–	–	
Variety, Value	*Interoperability*				+
Value	*Availability*		–		

Bruno Sena, Lina Garcés, Ana Allian, Elisa Y. Nakagawa. Investigating the applicability of architectural patterns in big data systems. In: Proceedings of the 25th Conference on pattern language of programs (PLoP), 2018.

A escolha de um padrão para beneficiar um atributo “quase sempre” leva a **conflictos** para abordar outros atributos. Alguns exemplos são:

Segurança x Desempenho
Manutenção x Segurança
Distribuição x Interoperabilidade
Modularidade x Eficiência
Confiabilidade x Portabilidade

É possível solucionar todos os conflitos em uma arquitetura?

É possível solucionar todos os conflitos em uma arquitetura?

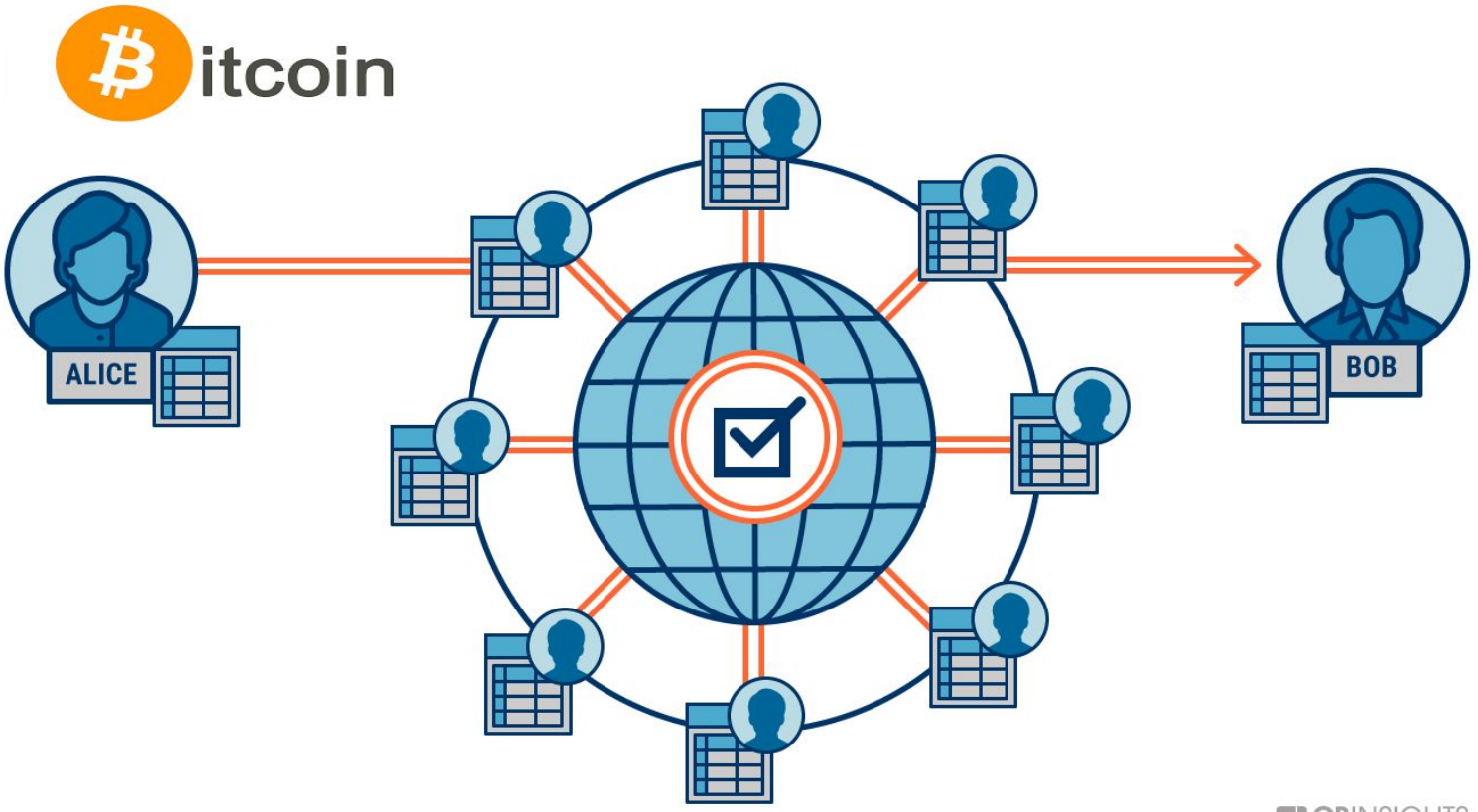
R: Às vezes **não** é possível.

Nessas situações devemos **priorizar** os requisitos de qualidade.

A falta de soluções para os requisitos com baixa prioridade é um tipo de **DÉBITO TÉCNICO** que **deve ser documentado**.

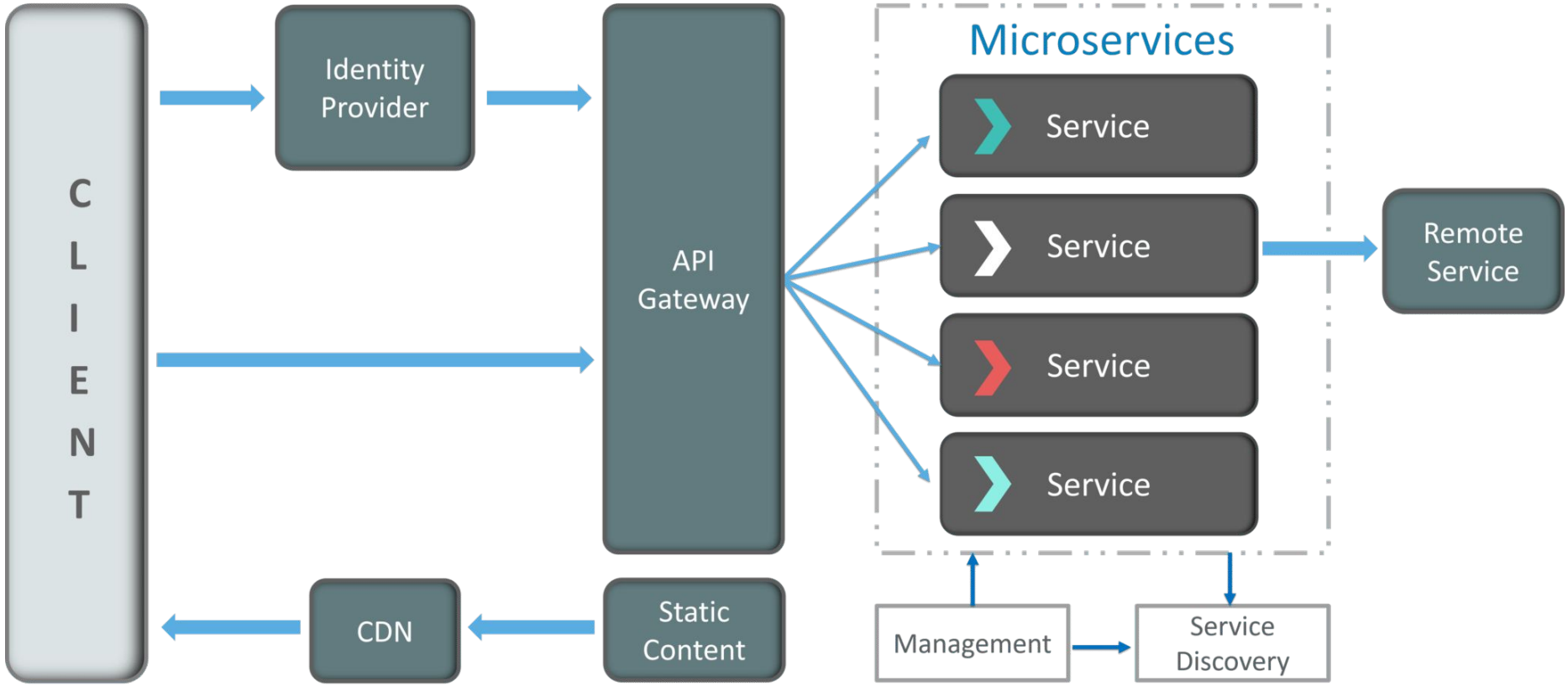
Padrões Recentes

Blockchain → Arquitetura base para o bitcoin



Padrões Recentes

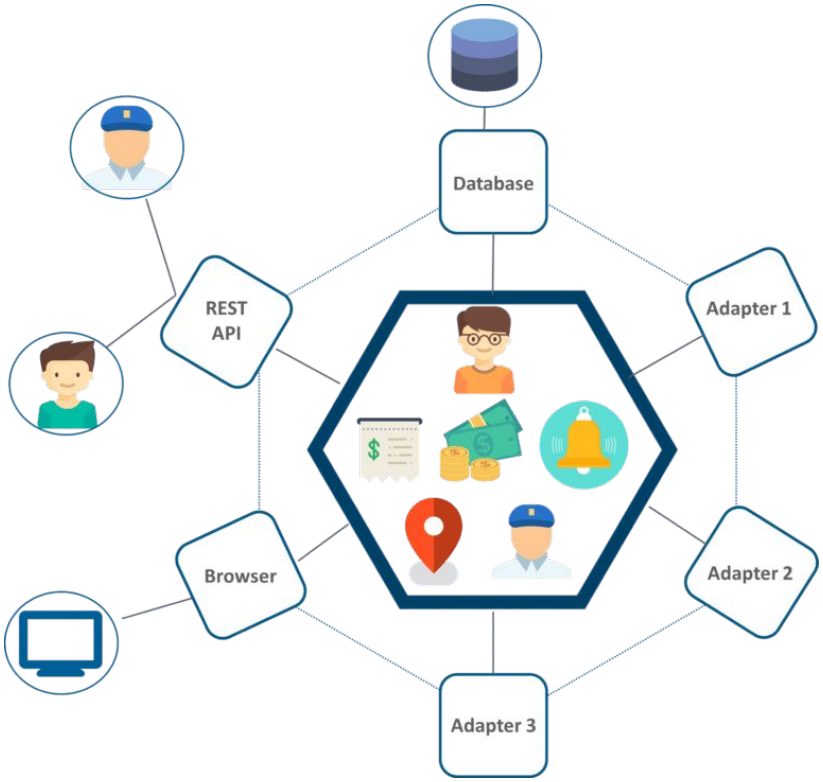
Microservices



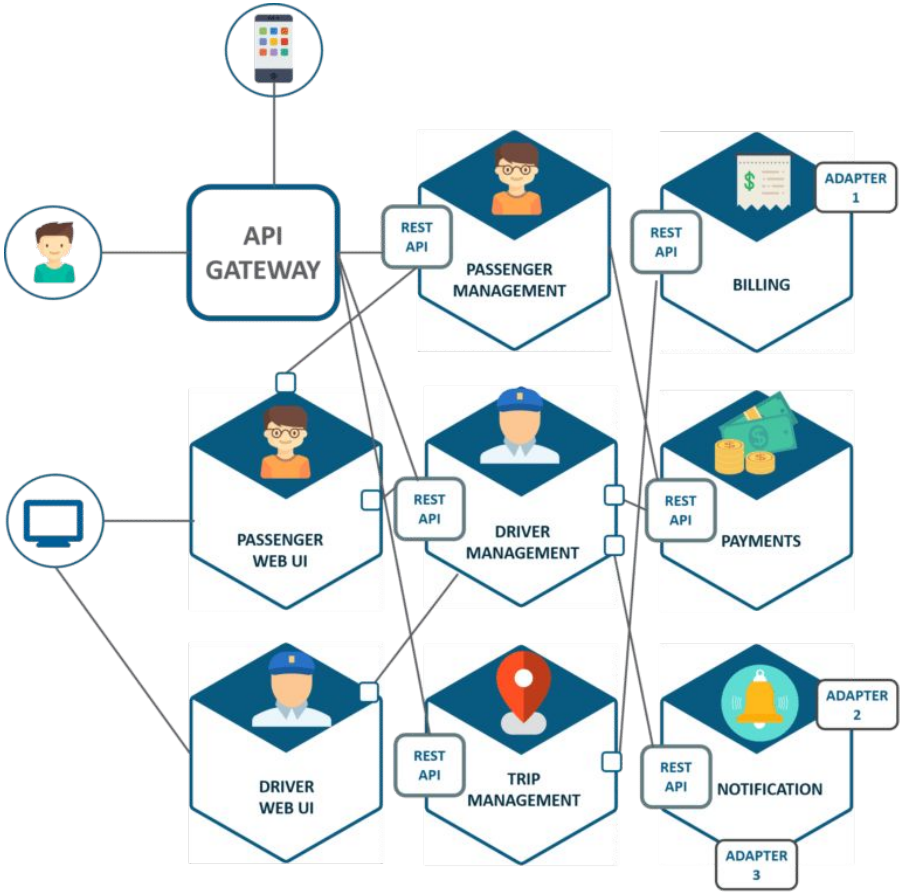
Content Delivery Networks (CDNs)

Padrões Recentes

Microservices



Arquitetura monolítica do Uber



Arquitetura em microserviços do Uber

Visões vs. Padrões Arquiteturais

- Padrões arquiteturais podem ser usados como ponto inicial para o projeto de uma arquitetura.

- Uma visão pode descrever (através de modelos) os padrões escolhidos para solucionar algum problema da arquitetura.

- O uso do padrão deve estar justificado

→ Porque o padrão X é o adequado para abordar o problema Y?

Referências

[Book] Mark Richards. 2015. Software Architecture Patterns Understanding Common Architecture Patterns and When to Use Them. pp. 54, O'Reilly Publishers.

[Book] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 1996. Pattern-Oriented Software Architecture - Volume 1: A System of Patterns. pp. 157. Wiley Publishing.

[Paper] Neil B. Harrison and Paris Avgeriou. Leveraging architecture patterns to satisfy quality attributes. ECSA, 2007.

[Paper] Paris Avgeriou and Uwe Zdun. Architectural patterns revisited - a pattern language. 2005.

Aula 9 - Padrões Arquiteturais

SSC0620/SSC5764 - Engenharia de Software
1º Semestre de 2019
Profa. Dra. Elisa Yumi Nakagawa e Prof. Dra. Lina Garcés
PAE: Brauner Oliveira e Daniel Santos

12 de Abril de 2019