Engenharia de Software

Arquitectura de Software
Padrões de Arquitectura de Software

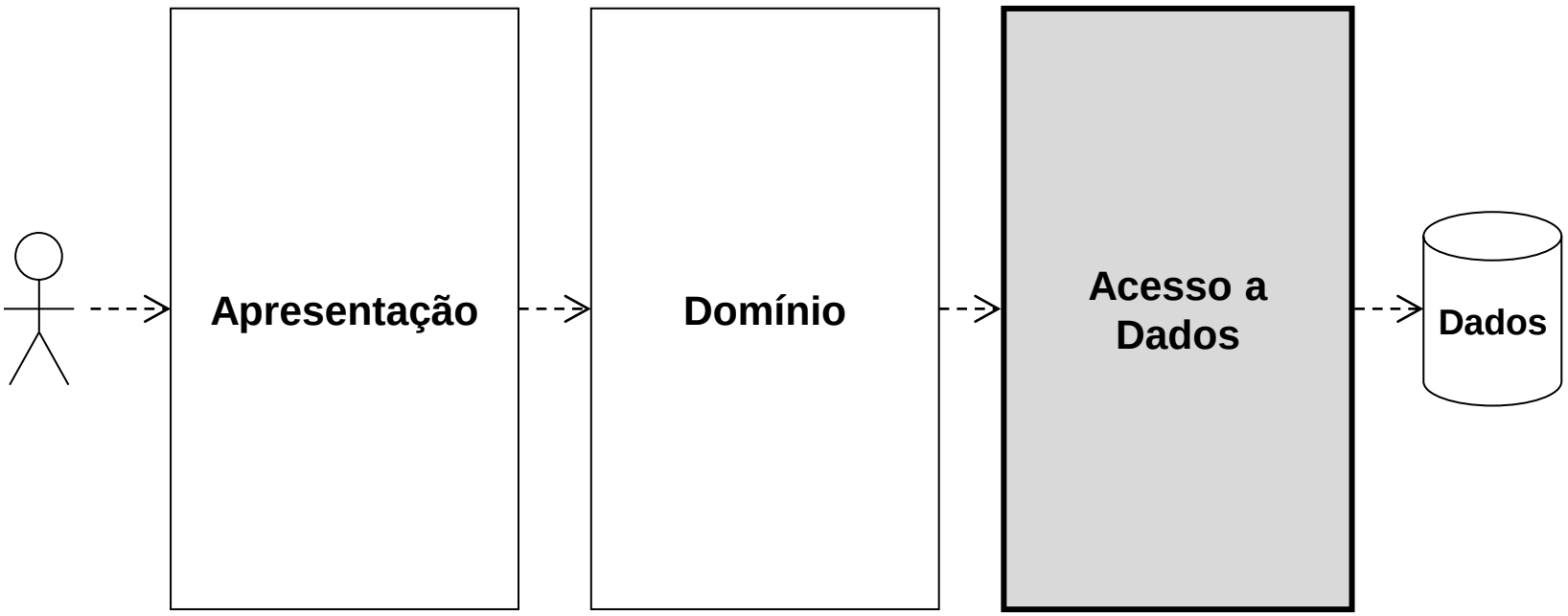
Luís Morgado

Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Arquitectura de 3 Camadas

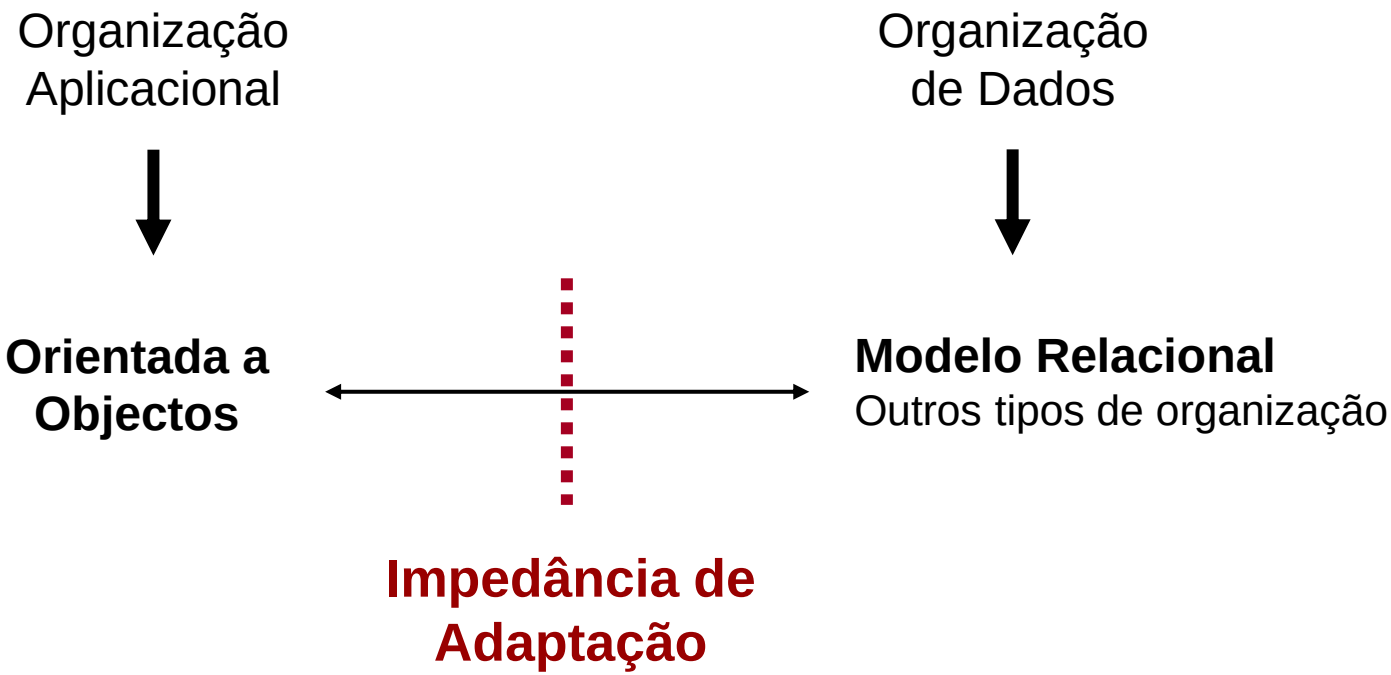
Camada de Acesso a Dados

A camada de acesso a dados é responsável pelo acesso e persistência dos dados, abstraindo os detalhes de acesso e armazenamento dos dados, de modo a fornecer uma interface simplificada para as restantes camadas do sistema



Arquitectura Aplicacional e de Dados

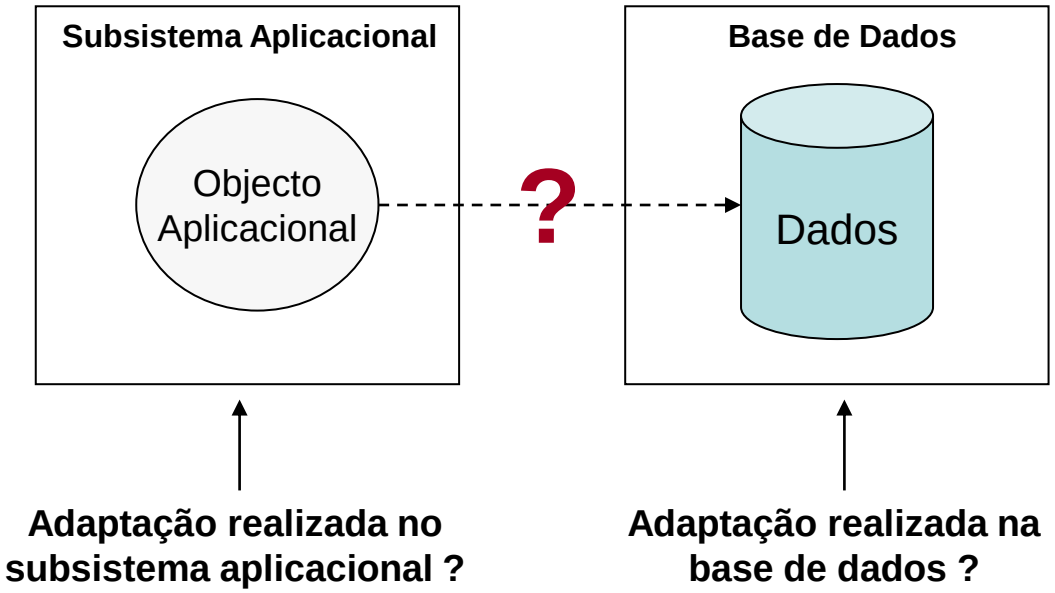
Relação entre *arquitectura aplicacional* e *arquitectura de dados*



A organização aplicacional dos dados, nomeadamente, orientada a objectos é, em geral, diferente da organização lógica dos dados, o que levanta um problema de adaptação de modelos, que pode ter complexidade significativa

Arquitectura Aplicacional e de Dados

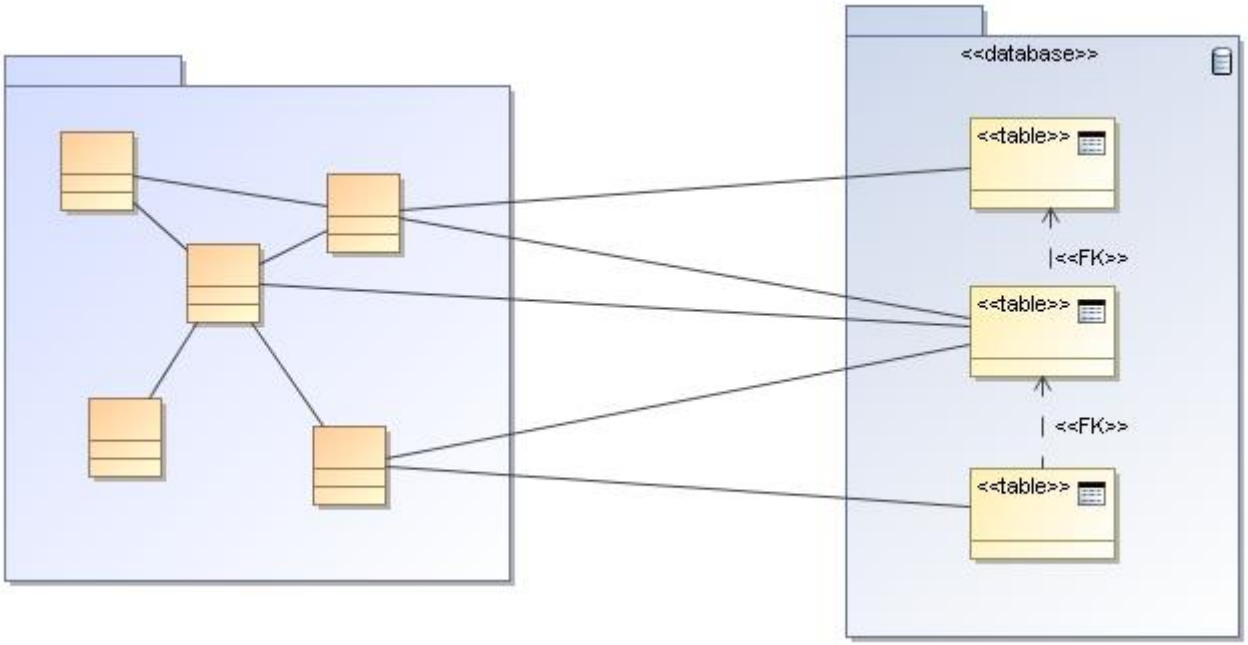
Relação entre *arquitectura aplicacional* e *arquitectura de dados*



A adaptação de modelos pode ser realizada quer a nível aplicacional, quer a nível da base de dados

Organização do Acesso a Dados

Independentemente da forma como a adaptação de modelos é realizada, é necessário salvaguardar elevada coesão e baixo acoplamento, perante a fragmentação de dados resultante da adequada organização do modelo de dados, nomeadamente, para evitar redundância e garantir a consistência dos dados, por exemplo, por meio de métodos de normalização de dados

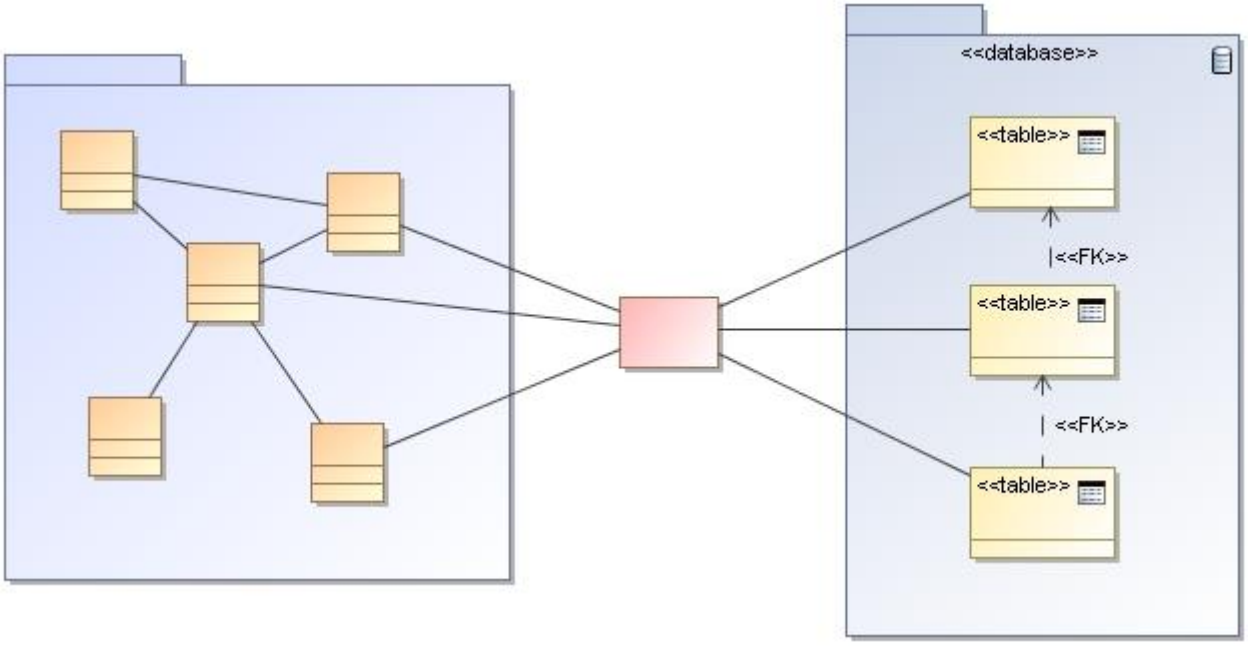


Organização Aplicacional

Organização Física

Organização do Acesso a Dados

Uma forma de realizar a adaptação de modelos é através da criação de um adaptador, que serve de intermediário entre a organização aplicacional e a organização física de dados



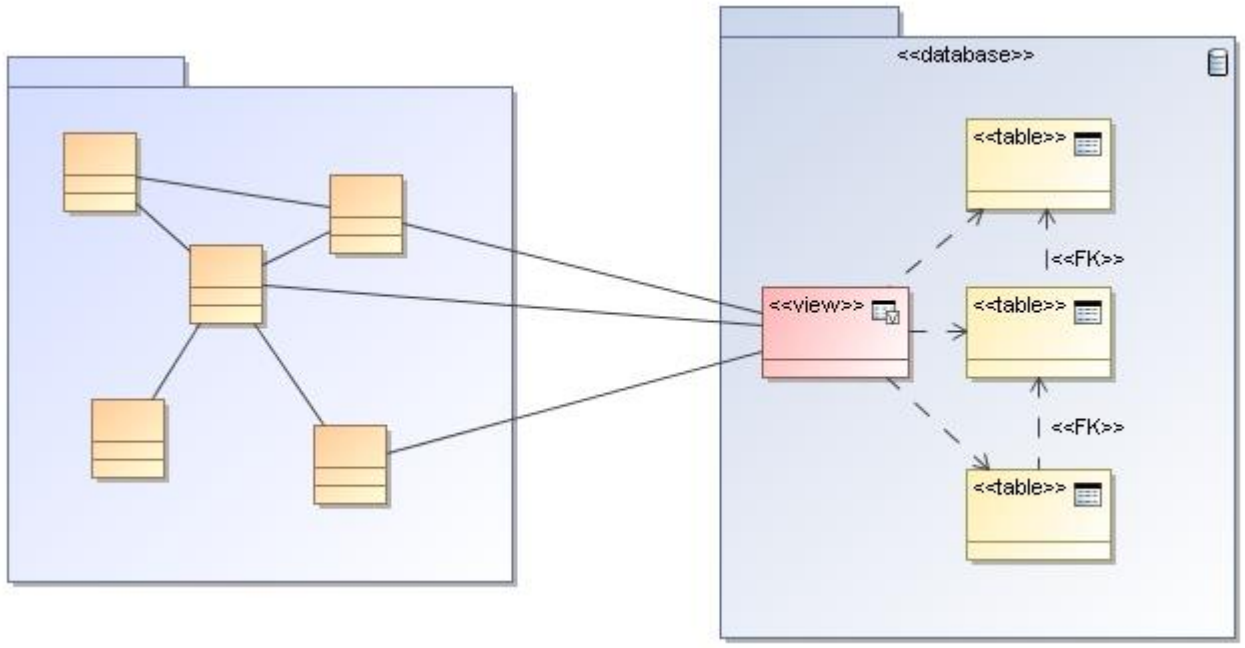
Organização Aplicacional

Adaptador

Organização Física

Organização do Acesso a Dados

O adaptador que serve de intermediário entre a organização aplicacional e a organização física de dados pode ser localizado junto aos dados, por exemplo, sob a forma de *vistas* de abstracção do acesso a dados



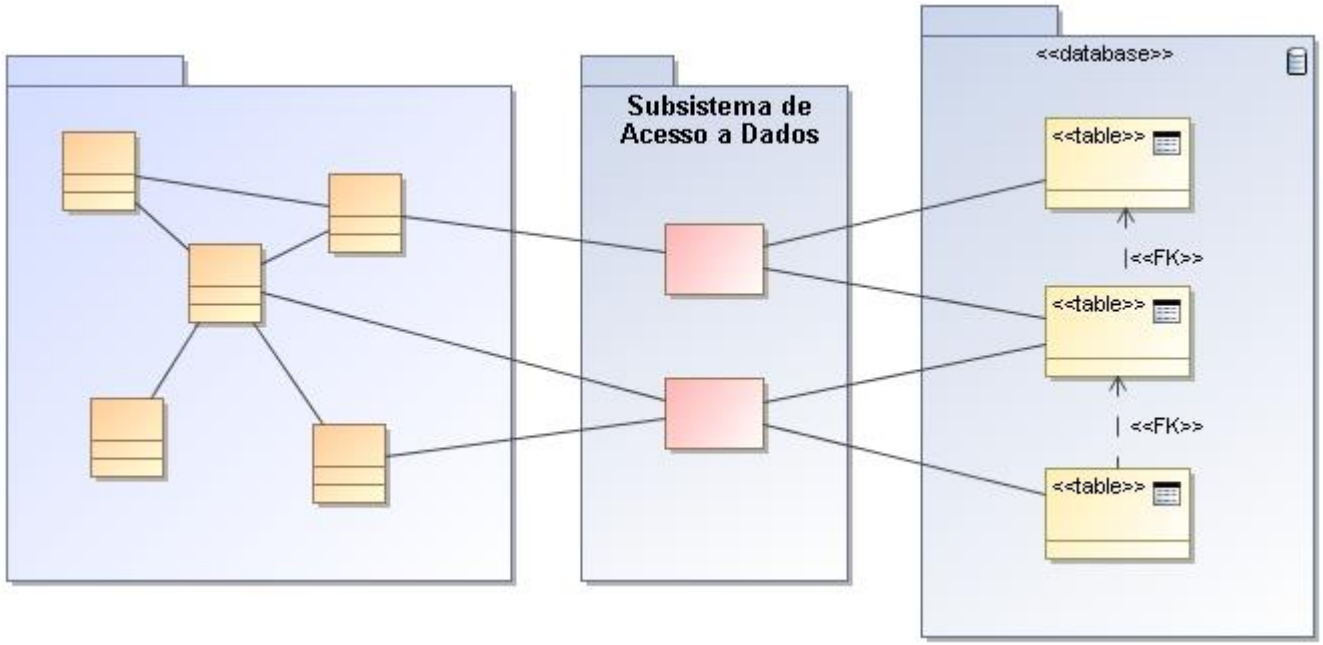
Organização Aplicacional

Adaptador

Organização Física

Organização do Acesso a Dados

O adaptador que serve de intermediário entre a organização aplicacional e a organização física de dados, também pode ser localizado num subsistema específico de acesso a dados



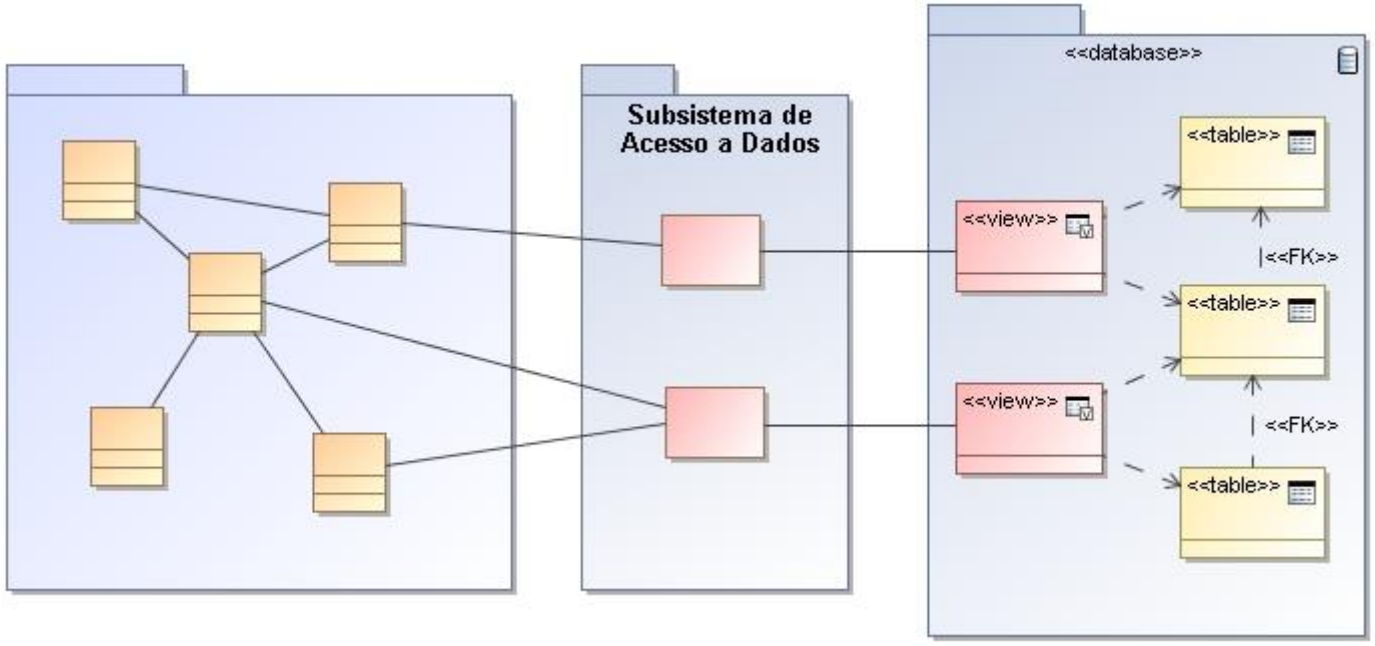
Organização Aplicacional

Adaptador

Organização Física

Organização do Acesso a Dados

Uma solução de grande flexibilidade e robustez para a adaptação de modelos, é a adaptação ser realizada por um subsistema de acesso a dados em conjunto com vistas de abstracção dos dados, localizadas junto aos dados



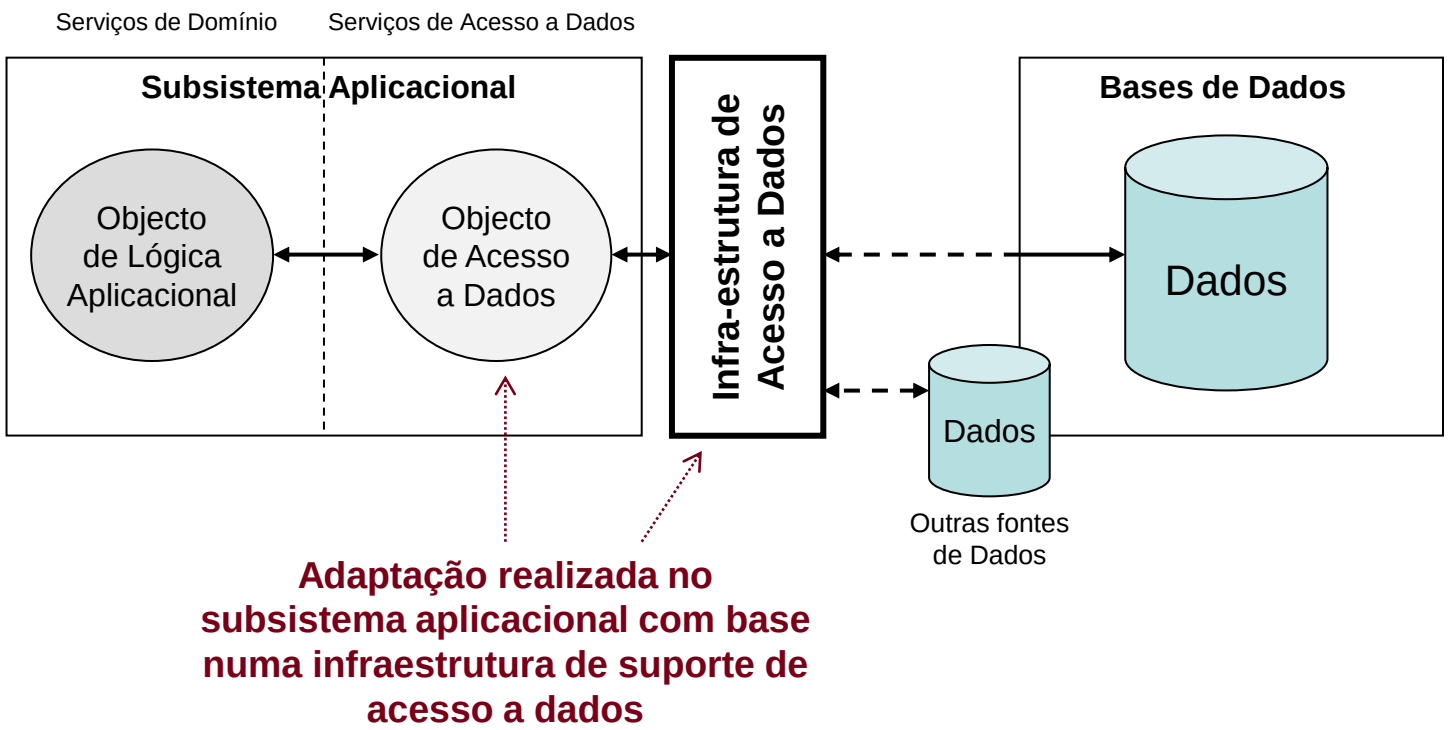
Organização Aplicacional

Adaptador

Organização Física

Arquitectura de Acesso a Dados

Relação entre arquitectura aplicacional e arquitectura de dados

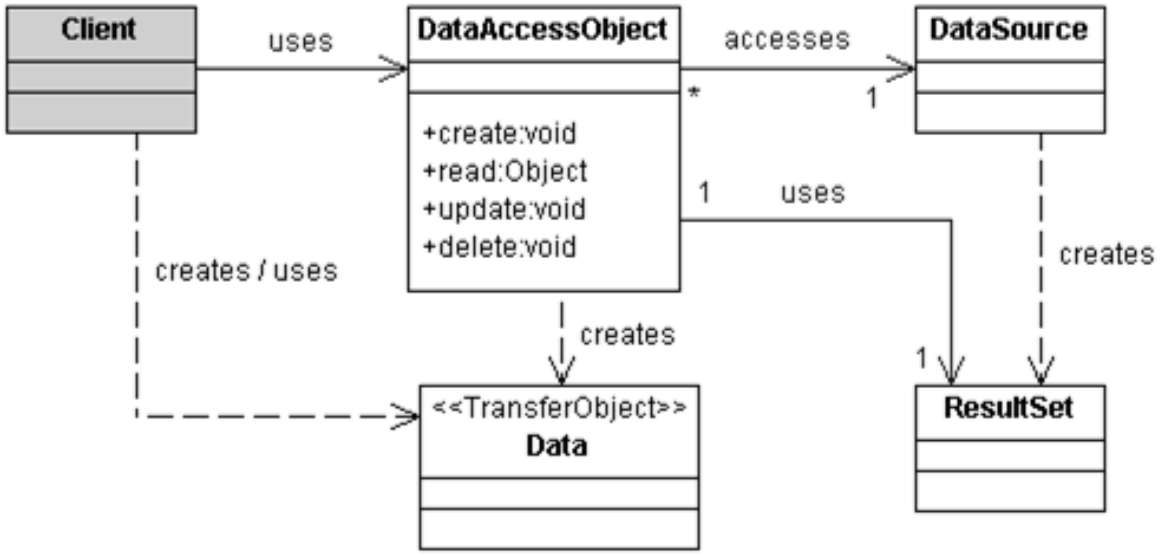


Padrão DAO (*Data Access Object*)

- **Problema**
  - Necessidade de reduzir o acoplamento entre a camada de domínio e a camada de acesso a dados, nomeadamente, de modo a ser possível alterar a fonte de dados subjacente sem afetar a classe de domínio
- **Solução**
  - Abstrair o acesso à fonte de dados (base de dados, ficheiro, etc.) através de uma interface de abstracção da fonte de dados
- **Consequências**
  - Redução de acoplamento entre as camada de domínio e a camada de acesso a dados por encapsulamento dos detalhes de acesso a dados, de forma que as classes de domínio não necessitam de conhecer os detalhes de interacção com a fonte de dados
  - Permite mudar a fonte de dados ou a forma de persistência de dados sem que isso influencie a camada de domínio
  - Promove a eliminação de redundância no acesso a dados
  - Contribui para simplificar o acesso a dados

Padrão DAO (*Data Access Object*)

Estrutura

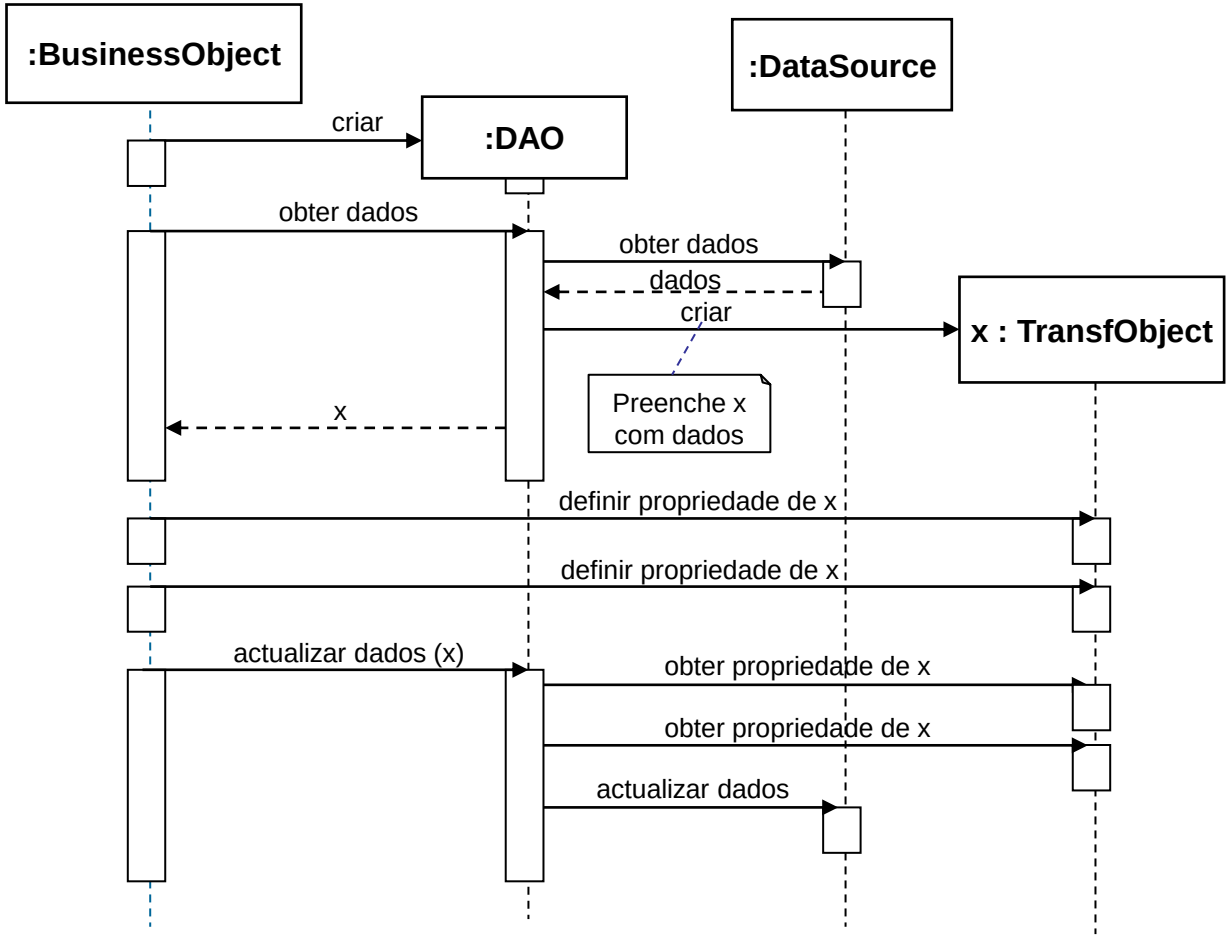


Participantes

- *DataAccessObject*
  - Responsável pelo acesso a dados
- *TransferObject*
  - Suporta a transferência de dados entre a camada de domínio e a camada de acesso a dados
- *DataSource*
  - Fonte de dados, abstrai o acesso ao suporte de persistência de dados
- *ResultSet*
  - Estrutura de dados resultante do acesso à fonte de dados, serve de base à criação do *TransferObject*

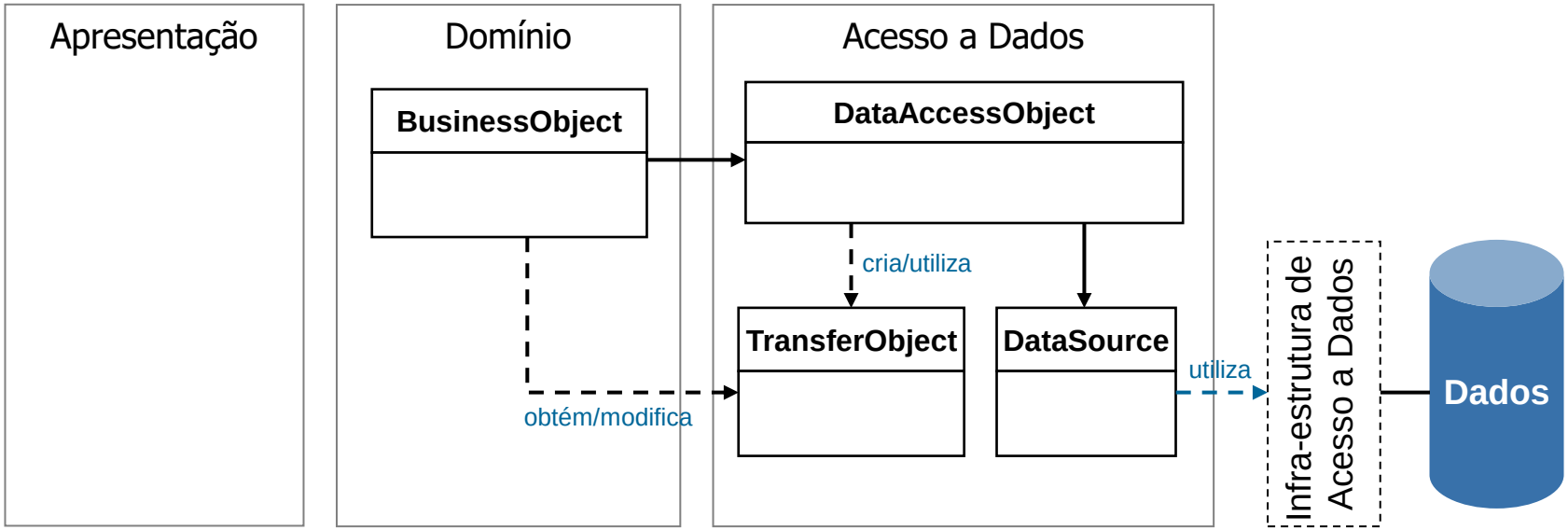
Padrão DAO (*Data Access Object*)

Comportamento



Padrão DAO (*Data Access Object*)

Enquadramento do padrão DAO no contexto de uma arquitectura aplicacional de 3 camadas

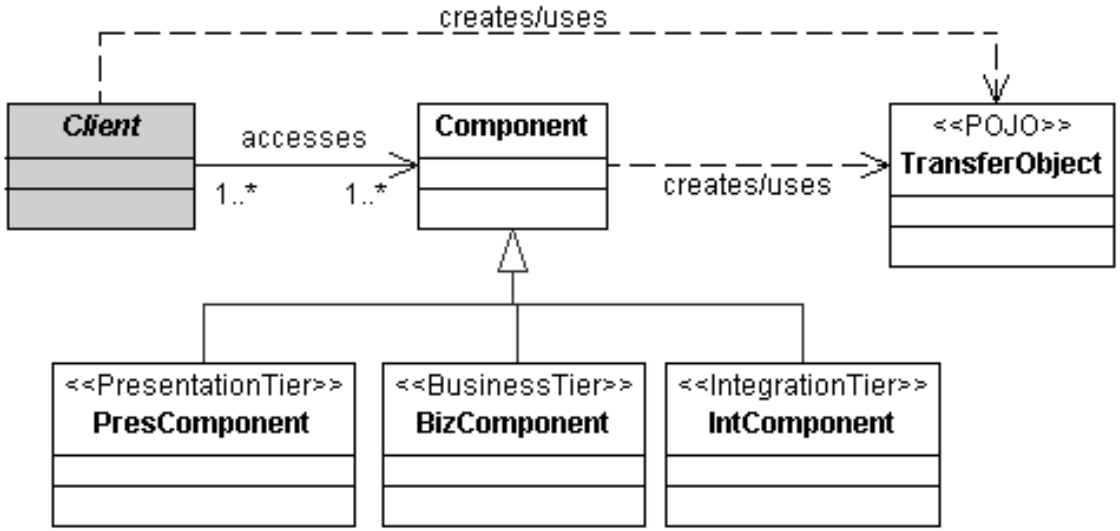


Padrão Objecto de Transferência (*Transfer Object*)

- **Problema**
  - Necessidade de evitar múltiplas interacções entre camadas para transferir dados relacionados
- **Solução**
  - Agregar os dados num único objecto que é utilizado para realizar a transferência de dados entre camadas
- **Consequências**
  - Simplifica a lógica de transferência de dados entre camadas
  - Transferência de mais dados com menos interacções entre camadas
  - Pode gerar um aumento da complexidade do sistema e do esforço de manutenção, devido à necessidade de definir múltiplos tipos de objectos de transferência

Padrão Objecto de Transferência (*Transfer Object*)

Estrutura



Participantes

- *Client*
  - Representa uma parte que interage com um componente de qualquer outra camada transferindo dados com base em objectos de transferência
- *Component*
  - Representa uma parte de qualquer camada que cria e utiliza objectos de transferência para enviar e receber dados entre objectos das mesma camada ou de outras camadas
- *TransferObject*
  - Suporta a transferência de dados entre camadas

Padrão Objecto de Transferência (*Transfer Object*)

- **Objecto de transferência (*Transfer Object*)**
  - Objecto que transporta dados entre subsistemas
  - Forma de reduzir o custo de comunicação entre subsistemas, reduzindo o número de interacções através da utilização de um objecto de transferência que agrega os dados, de modo a ser necessária apenas uma interacção para a transferência de um conjunto de dados relacionados
- **Comparação com objectos de domínio**
  - Um objecto de transferência não tem qualquer comportamento associado, excepto para acesso aos seus próprios dados
    - Tipicamente **imutável**
  - Um objecto de transferência não deve conter qualquer lógica de domínio

Arquitectura de 3 Camadas

Limitações

A arquitectura aplicacional de 3 camadas, apesar de apresentar características relevantes para lidar com a complexidade de aplicações em arquitecturas cliente-servidor, apresenta limitações, por exemplo, no contexto de integração de serviços prestados por diferentes aplicações, nomeadamente:

- Flexibilidade limitada
  - A arquitetura de 3 camadas tem uma estrutura fixa com três camadas distintas, apresentação, domínio e acesso a dados, o que pode limitar a flexibilidade para modificar camadas ou acrescentar camadas adicionais com base em requisitos específicos ou avanços tecnológicos
- Modularidade limitada
  - A organização em apenas 3 camadas pode criar limitações em termos da organização modular de uma aplicação complexa, a qual pode necessitar de mais camadas

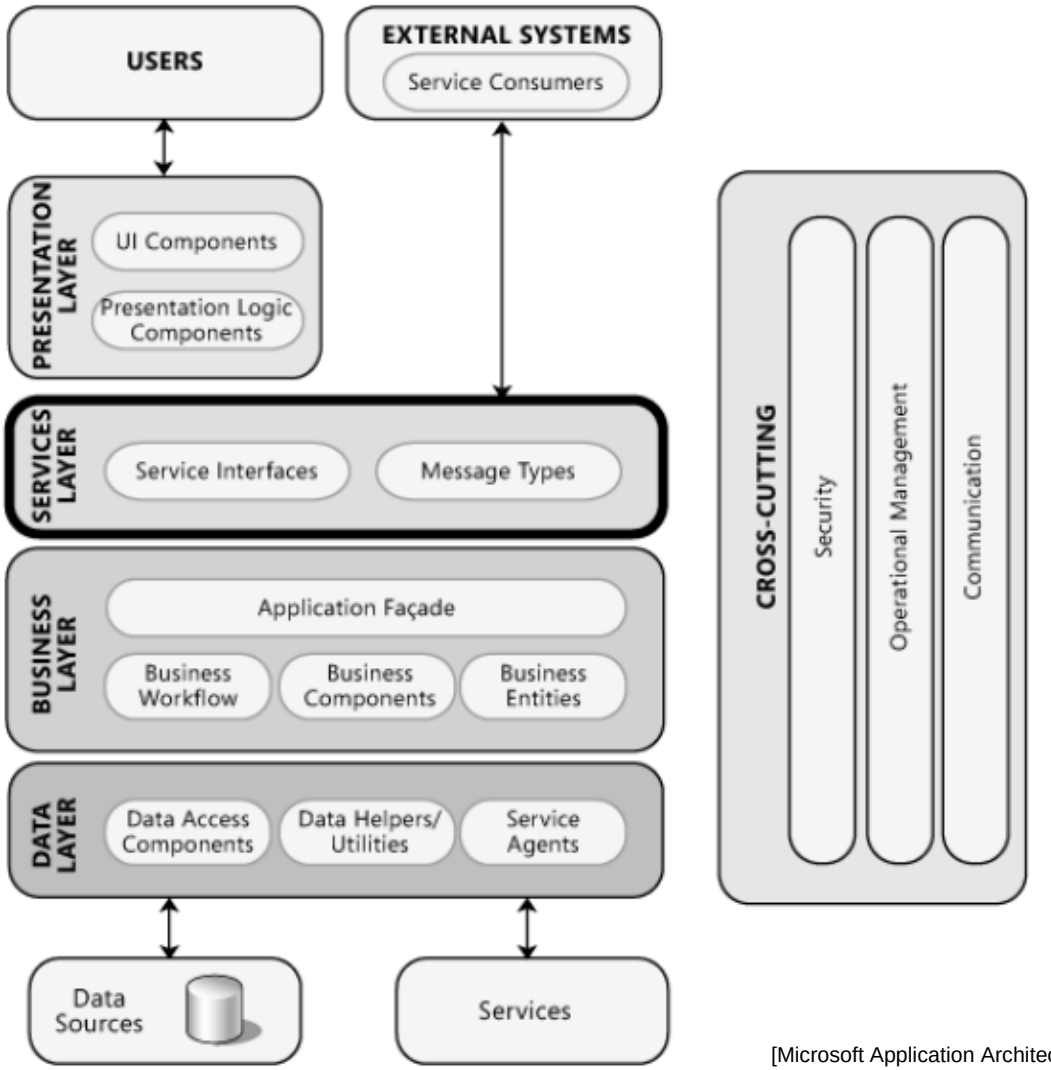
Arquitectura Aplicacional Multicamada

Arquitetura aplicacional multicamada é um padrão de arquitetura onde as partes de uma aplicação são organizadas não apenas em 3 camadas, mas em múltiplas camadas, sendo as principais diferenças em relação à arquitectura de 3 camadas as seguintes:

- Flexibilidade e escalabilidade
  - Permite maior flexibilidade e escalabilidade, pois podem ser adicionadas camadas adicionais para suportar novas funcionalidades ou serviços, bem como diferentes formas de implantação e disponibilização de serviços, nomeadamente, no que se refere a balanceamento de carga ou organização em contentores
- Separação de responsabilidades
  - Fornece um nível mais elevado de separação de responsabilidades, dividindo a aplicação em diferentes camadas, cada uma responsável por um aspecto específico da aplicação
- Gestão de complexidade
  - Possibilita o desenvolvimento de aplicações mais complexas, através da distribuição de responsabilidades por diferentes camadas

Arquitectura Multicamada

Organização geral

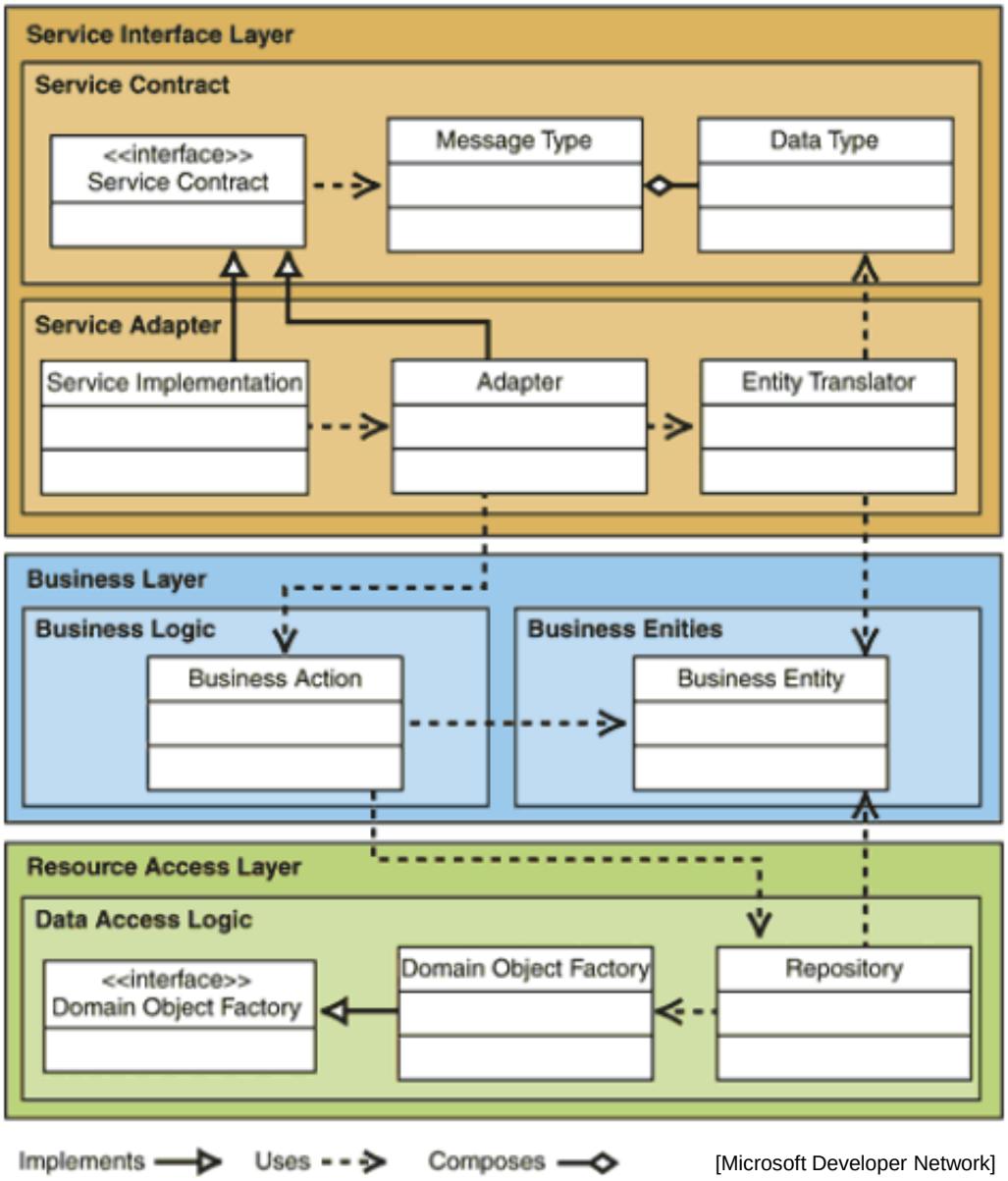


A arquitetura aplicacional multicamada é organizada em múltiplas camadas, de acordo com o padrão *Camadas* (*Layers*) podendo, no entanto, incluir camadas transversais de suporte às restantes camadas

Este tipo de arquitectura está particularmente vocacionada para suportar a integração de serviços disponibilizados por diferentes aplicações ou por outros prestadores de serviços

[Microsoft Application Architecture Guide, 2nd Ed., 2009]

Arquitectura Multicamada



[Microsoft Developer Network]

Numa arquitetura aplicacional multicamada, cada camada pode ser subdividida em subcamadas responsáveis por aspectos específicos dessa camada

Se necessário, uma camada de interface de serviços define os contractos de serviço e os adaptadores necessários para integração de serviços

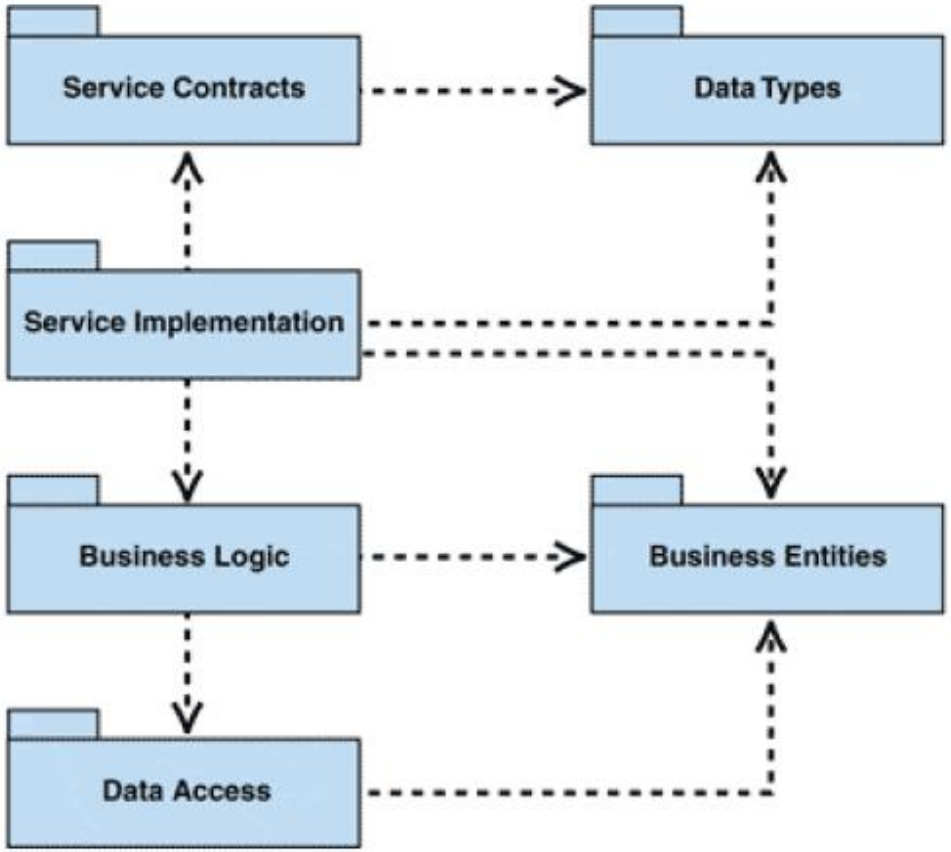
A camada de domínio (*negócio*) é tipicamente dividida em objectos de serviço (*acções de negócio*) que realizam casos de utilização, e entidades (*entidades de negócio*) que realizam as entidades do domínio do problema

A camada de acesso a dados da arquitectura de 3 camadas pode ser substituída por uma *camada de acesso a recursos*, a qual abstrai o acesso a recursos locais ou remotos, ou a utilização de serviços disponibilizados por outros sistemas

Arquitectura Multicamada

De modo a garantir a redução do acoplamento entre camadas, evitando dependências bidireccionais entre camadas, as camadas são organizadas com uma separação de responsabilidades mais específica, aumentando assim a coesão e modularidade da arquitectura

Em particular, a camada de domínio (*business layer*) é divida numa camada de lógica de domínio (*business logic*) e numa camada de entidades de domínio (*business entities*)



[Microsoft Developer Network]

Objectos e Entidades de Domínio

- **Objectos de domínio (*Business Objects*)**
  - Objectos concebidos principalmente em torno das responsabilidades e dos comportamentos definidos por um caso de utilização
  - Tem por objectivo realizar o comportamento descrito nos casos de utilização em termos da lógica de domínio, bem como garantir as restrições de domínio, como regras de validação e regras de autorização
- **Entidades de domínio (*Business Entities*)**
  - Objectos concebidos principalmente como contentores de dados
  - Suportam a lógica de acesso e persistência de dados em bases de dados (ou outros suportes de persistência de dados) em termos de entidades de domínio
- **Necessidade de garantia de consistência e integridade dos dados processados**
  - **Mecanismos transaccionais**

Padrão Transacção ACID

- **Problema**
  - Necessidade de garantir a consistência e integridade de unidades de processamento envolvendo múltiplas operações e elementos de dados
- **Solução**
  - Agregar o processamento numa transação com as propriedades ACID: *atomicidade*, *consistência*, *isolamento* e *durabilidade*
- **Consequências**
  - O resultado de uma transação é bem definido e previsível, garantindo a consistência e integridade dos dados processados
  - A utilização do padrão de transação ACID pode aumentar substancialmente a quantidade de armazenamento exigida, devido à necessidade de armazenar o estado inicial de cada objeto envolvido numa transação, para que, se a transação falhar, seja possível restaurar o estado inicial dos objectos envolvidos
  - Pode originar uma maior complexidade aplicacional devido à necessidade de gerir os mecanismos transaccionais

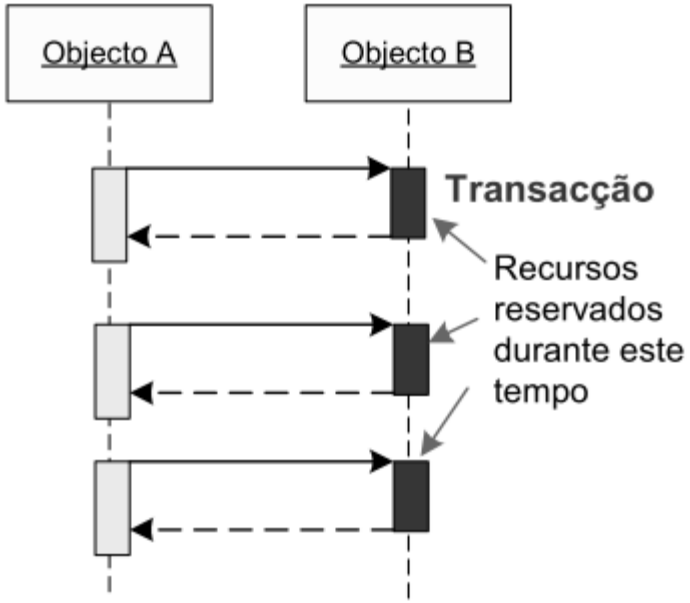
Mecanismos Transaccionais

Uma *transação* define uma unidade única lógica de processamento composta por uma ou mais operações, com garantia das seguintes propriedades:

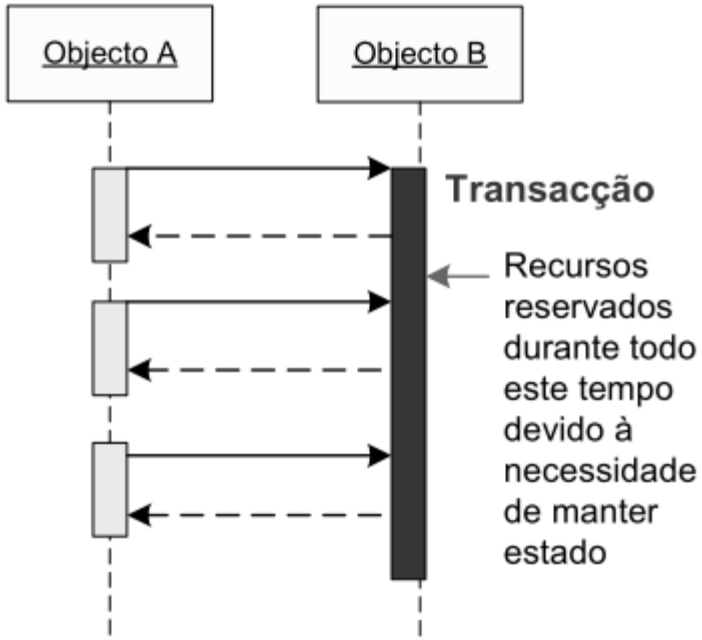
- **Atomicidade**
  - **Uma transacção é uma unidade de processamento atómica**, o que corresponde a uma característica ***tudo ou nada*** – todas as acções são realizadas com sucesso ou nenhuma é realizada
- **Consistência**
  - Ao realizar uma transacção **um sistema deve passar de um estado consistente para outro estado consistente** – não podem ser violadas quaisquer regras de integridade sobre os dados manipulados
- **Isolamento**
  - Os **efeitos das acções** que compõem uma transacção **só são visíveis após a transacção concluída com sucesso**
- **Durabilidade**
  - **Os efeitos de uma transacção terminada com sucesso são permanentes, mesmo em caso de falha do sistema** – caso em que, após a sua recuperação, este deve reflectir os resultados das transacções anteriormente concluídas com sucesso

Mecanismos Transaccionais

Transacções e manutenção de estado em arquitecturas aplicacionais



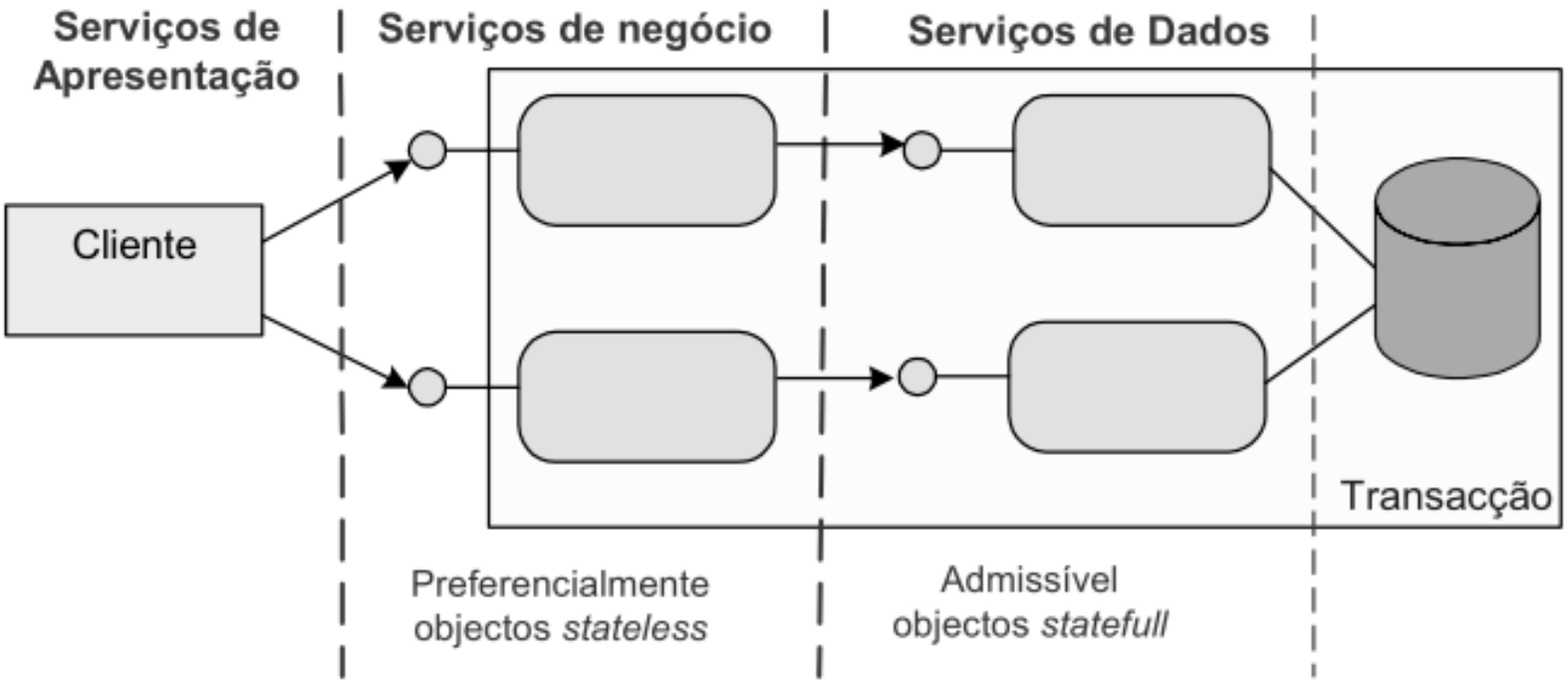
Comportamento sem manutenção de estado (*stateless*)



Comportamento com manutenção de estado (*stateful*)

Mecanismos Transaccionais

Transacções e manutenção de estado em arquitecturas aplicacionais



Bibliografia

[Pressman, 2003]

R. Pressman, *Software Engineering: a Practitioner's Approach*, McGraw-Hill, 2003.

[Gamma et al., 1995]

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[Shaw & Garlan, 1996]

M. Shaw, D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.

[Dijkstra, 1968]

E. Dijkstra, *The Structure of the 'THE' Multiprogramming System*, Communications of the ACM 11 – 5, 1968.

[Parnas, 1972]

D. Parnas, *On the Criteria to Be Used in Decomposing Systems into Modules*, Communications of the ACM 15-12, 1968.

[Kruchten, 1995]

F. Kruchten, *Architectural Blueprints - The "4+1" View Model of Software Architecture*, IEEE Software, 12-6, 1995.

[Grand, 1999]

M. Grand, *Transaction Patterns: A Collection of Four Transaction Related Patterns*. 6th Conference on the Pattern Languages of Programs (PLoP99), 1999.

[Booch, 2004]

G. Booch, *Software Architecture*, IBM, 2004.