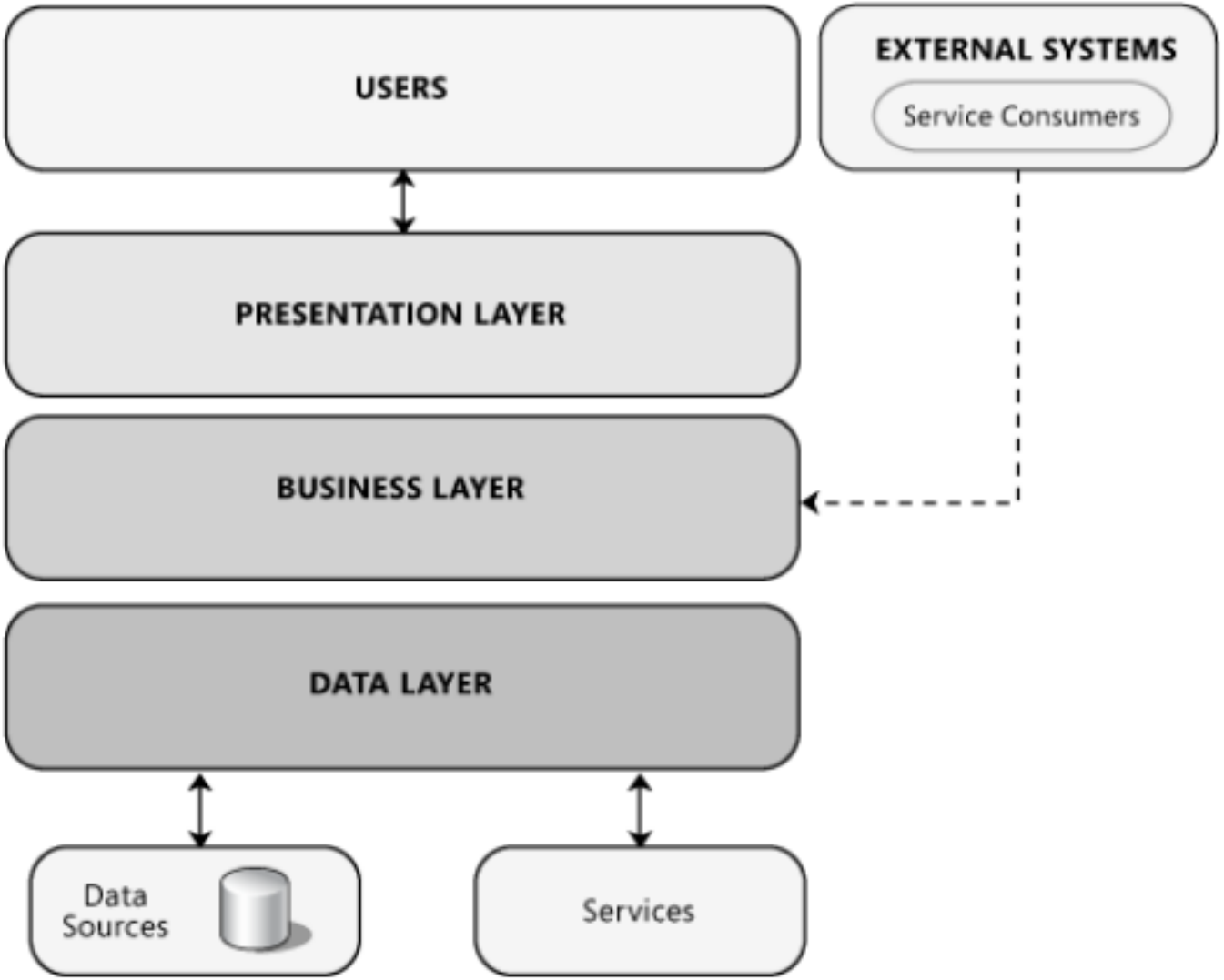
Engenharia de Software

Arquitectura de Software
Arquitecturas Multi-Camada

Luís Morgado

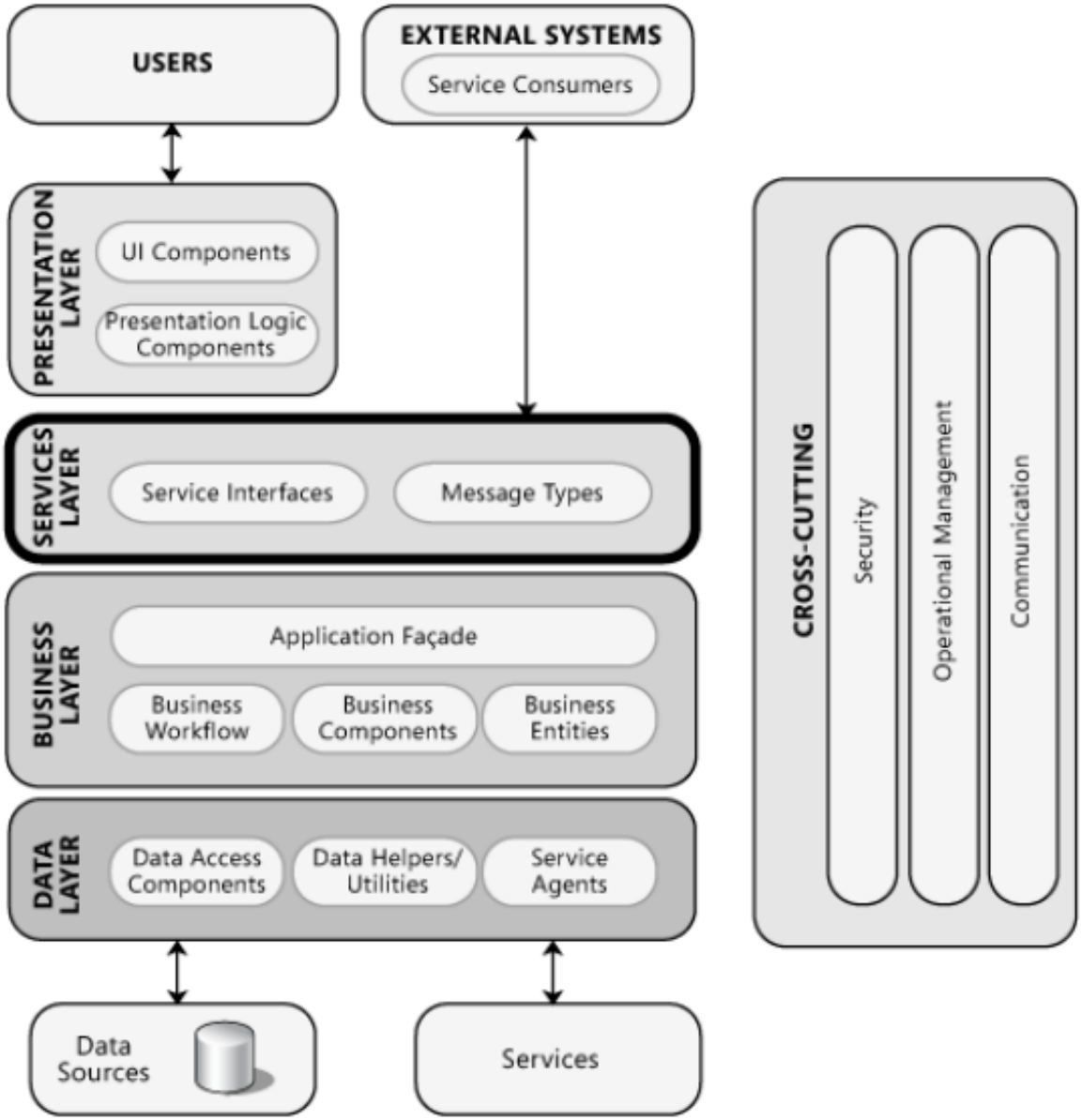
Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Arquitectura Multi-Camada



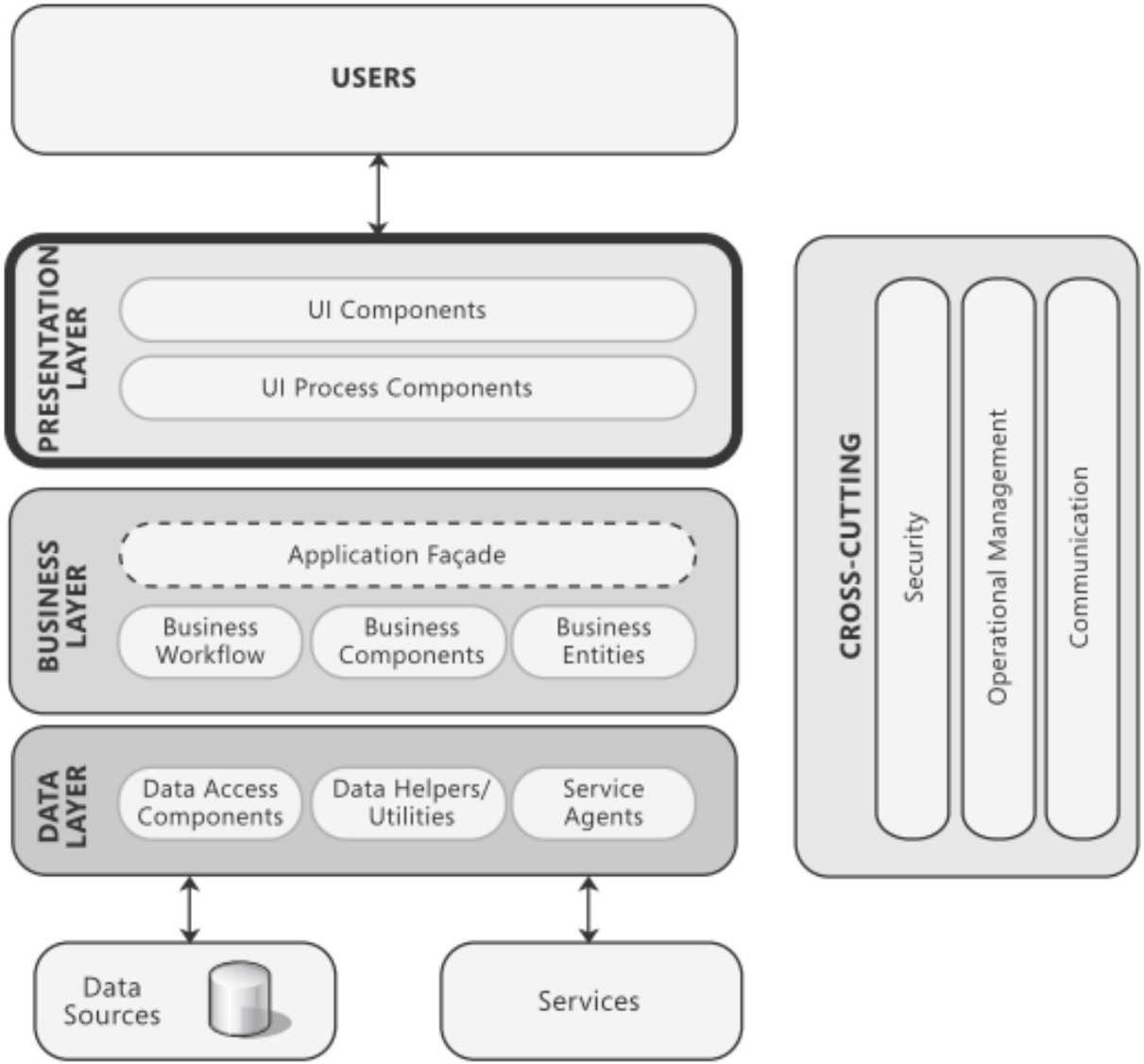
[Microsoft, 2009]

Arquitectura Multi-Camada



[Microsoft, 2009]

Mecanismos de Apresentação



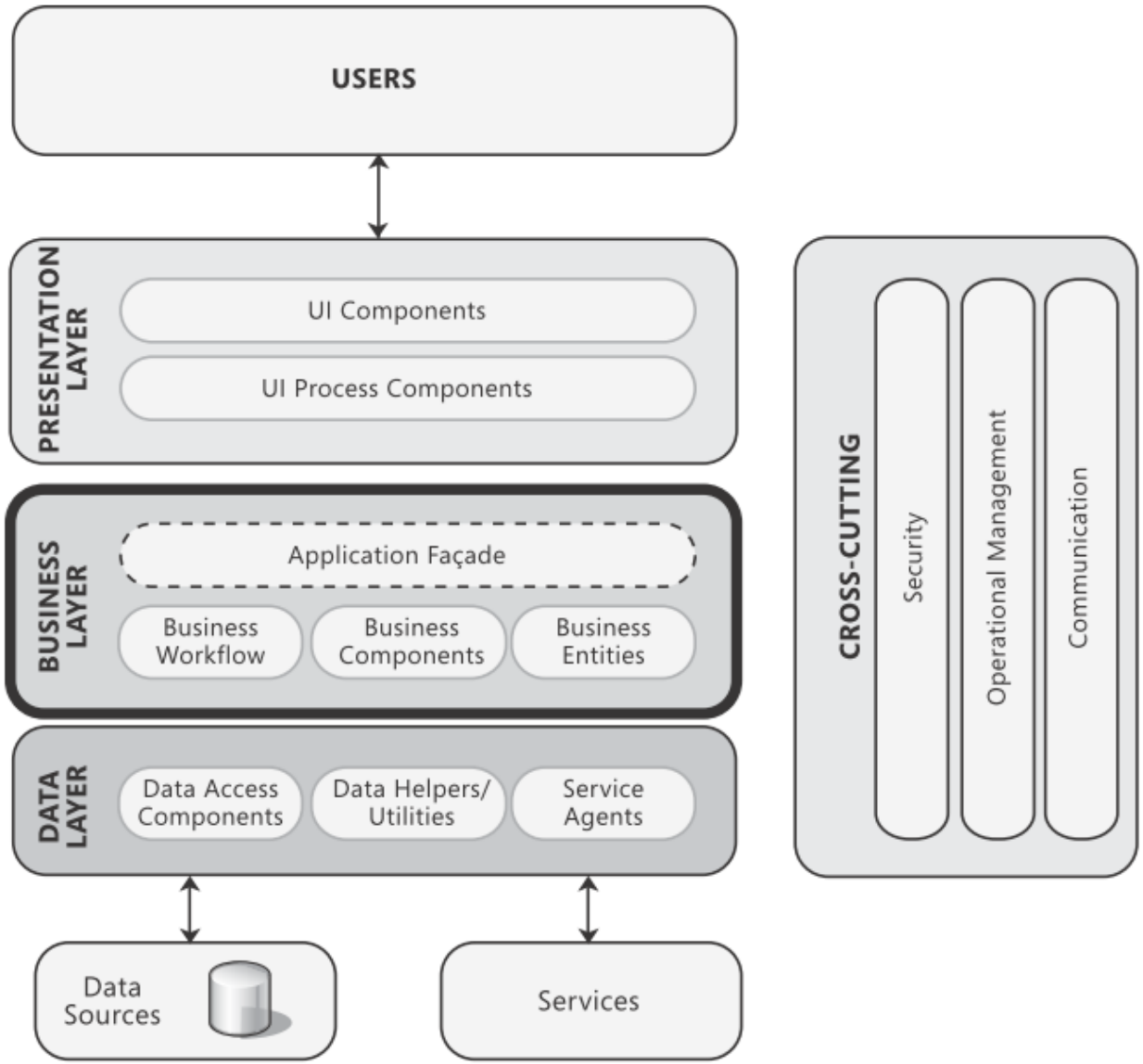
[Microsoft, 2009]

Mecanismos de Apresentação

Category	Relevant patterns
Composition and Layout	**Composite View.** Combine individual views into a composite representation. **Presentation Model.** (Model-View-ViewModel) pattern. A variation of Model-View-Controller (MVC) tailored for modern UI development platforms where the View is the responsibility of a designer rather than a classic developer. **Template View.** Implement a common template view, and derive or construct views using this template view. **Transform View.** Transform the data passed to the presentation tier into HTML for display in the UI. **Two-Step View.** Transform the model data into a logical presentation without any specific formatting, and then convert that logical presentation to add the actual formatting required.
Exception Management	**Exception Shielding.** Prevent a service from exposing information about its internal implementation when an exception occurs.
Navigation	**Application Controller.** A single point for handling screen navigation. **Front Controller.** A Web only pattern that consolidates request handling by channeling all requests through a single handler object, which can be modified at run time with decorators. **Page Controller.** Accept input from the request and handle it for a specific page or action on a Web site.

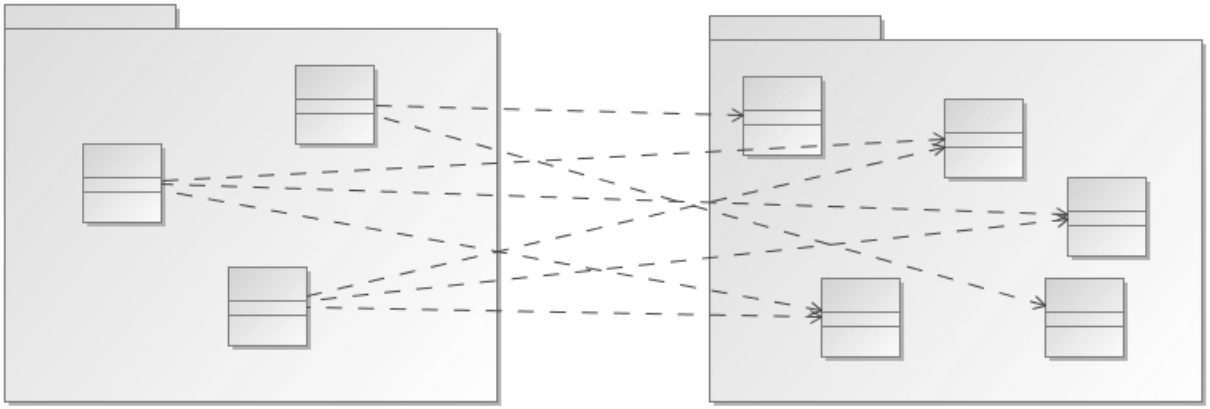
[Microsoft, 2009]

Mecanismos de Negócio

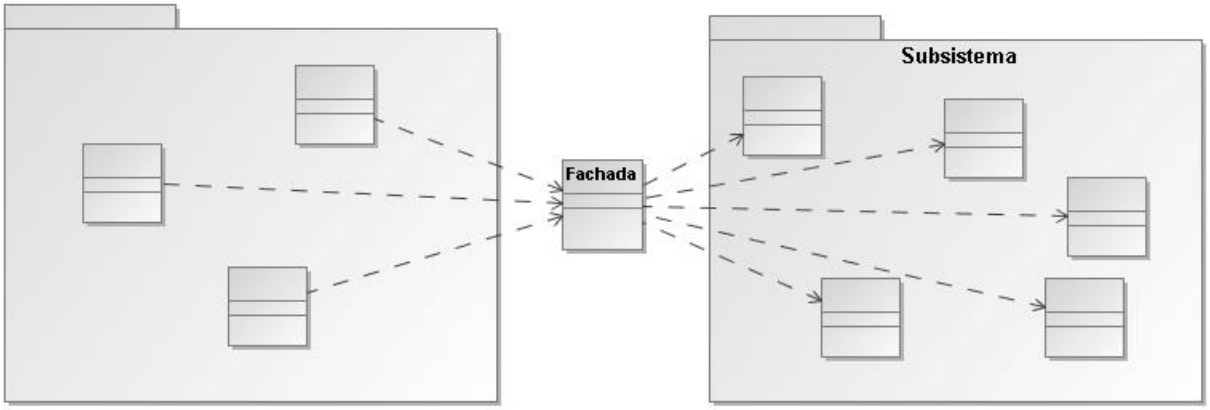


[Microsoft, 2009]

Redução de Acoplamento



Como melhorar arquitectura ?



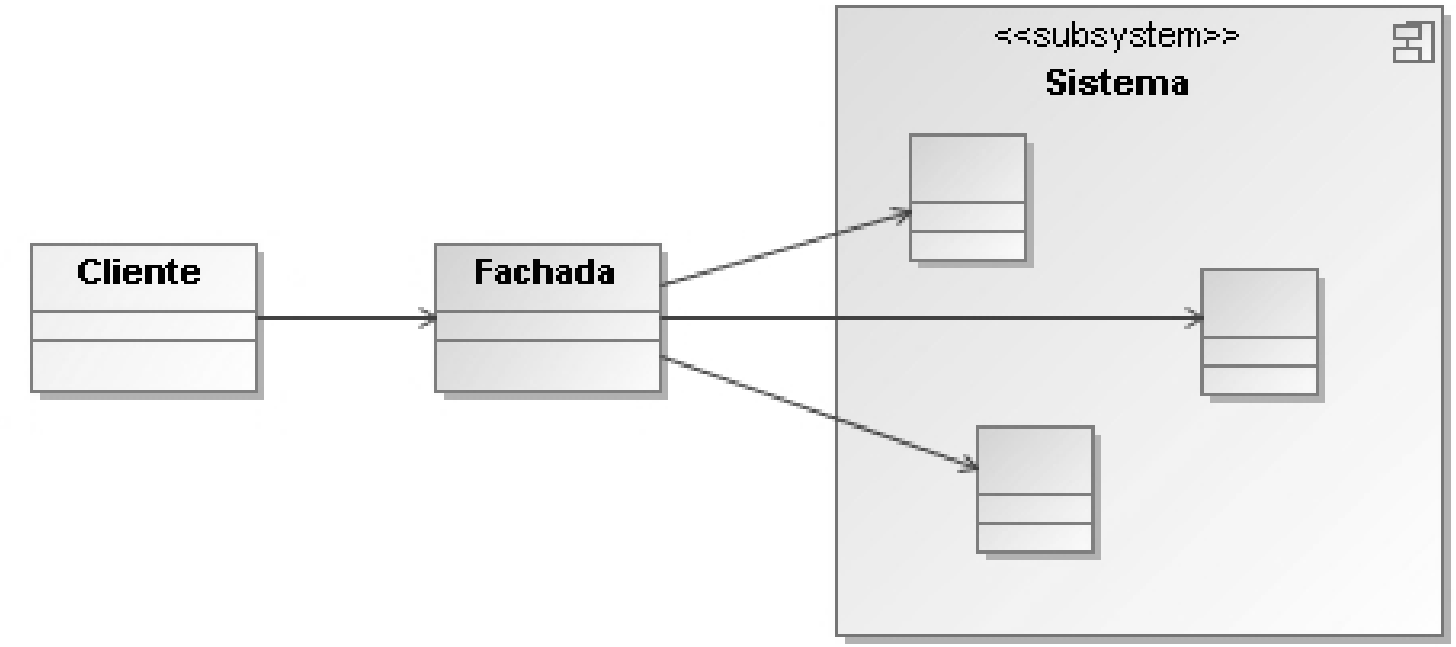
Redução de Acoplamento

- **Padrão Fachada (*Façade*)**
  - **Problema**
    - Dificuldade de **interacção com interfaces múltiplas e complexas**
  - **Solução**
    - Disponibilizar uma **interface uniforme** a partir de um conjunto de interfaces internas de um sistema
  - **Consequências**
    - **Encapsular um subsistema complexo** com base num objecto mais simples
    - Permite **reduzir o esforço de aprendizagem** para utilização do subsistema
    - Promove a **redução do nível de acoplamento**
    - Pode **limitar a flexibilidade** de utilização do sistema

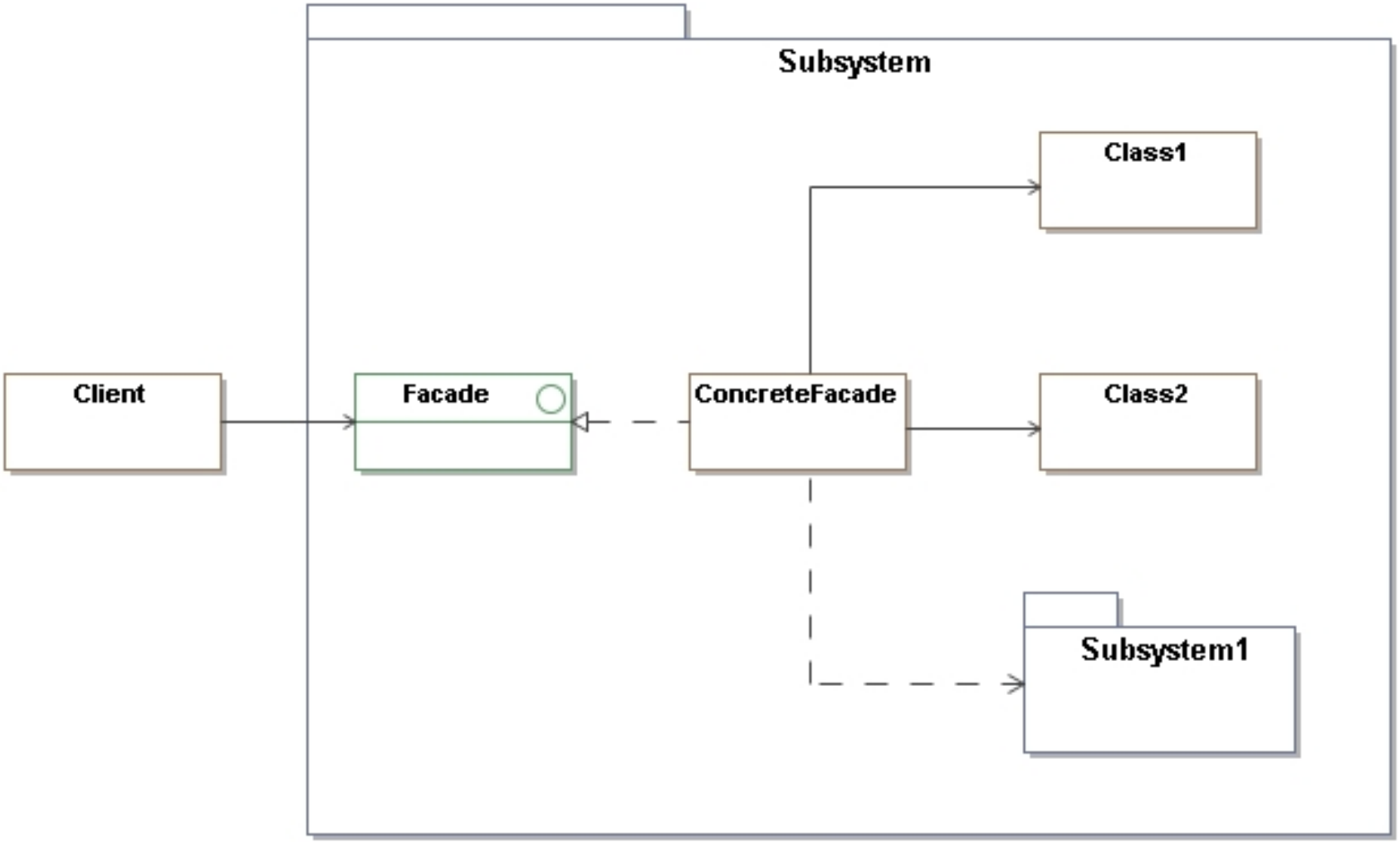
Padrão *Fachada* (*Façade*)

Aplicação de princípios de arquitectura

- Princípio do encapsulamento
- Princípio da minimização de acoplamento



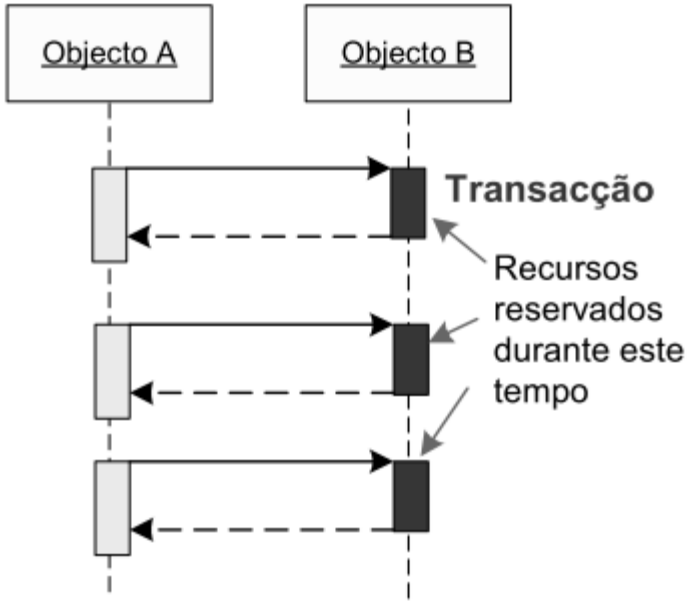
Padrão *Fachada* (*Façade*)



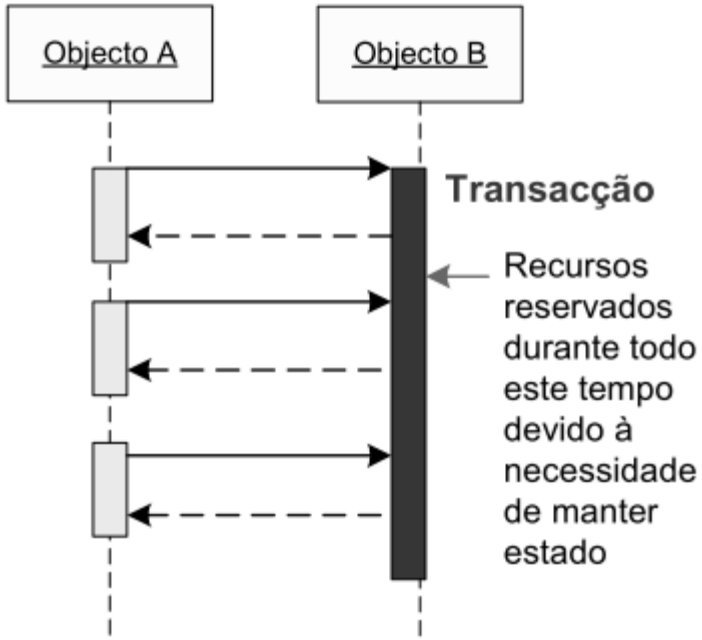
Transacções

- **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.

Transacções e Manutenção de Estado

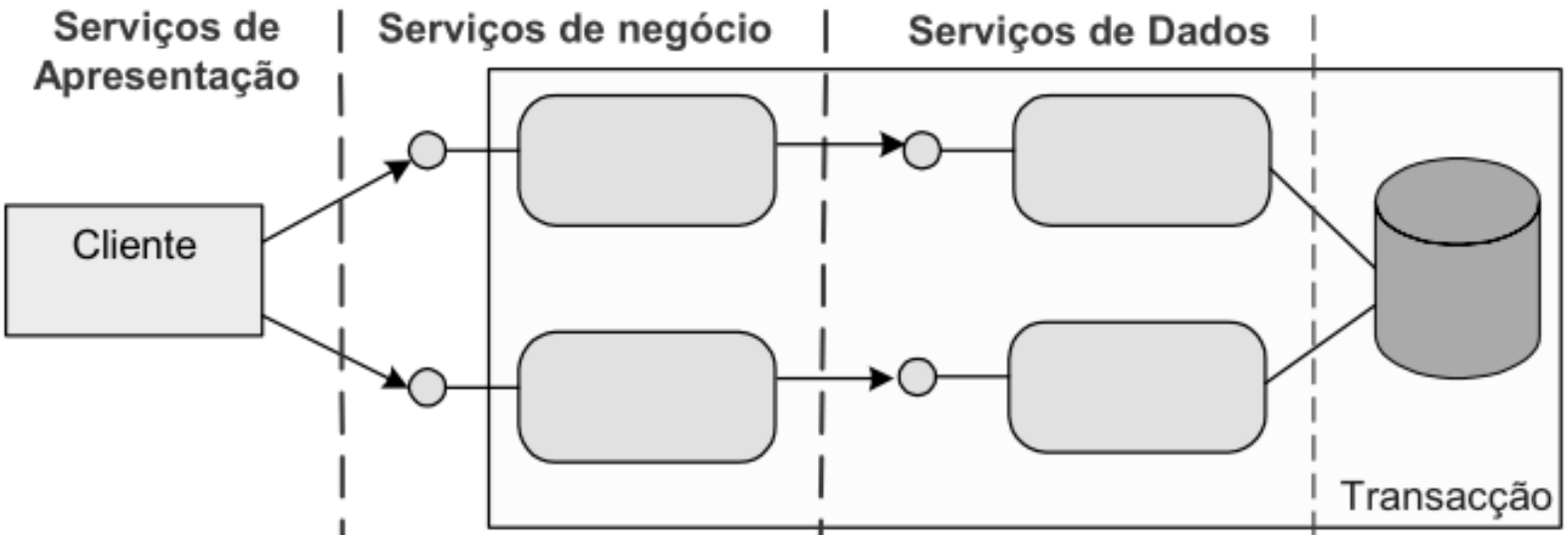


Comportamento sem manutenção de estado (*stateless*)



Comportamento com manutenção de estado (*stateful*)

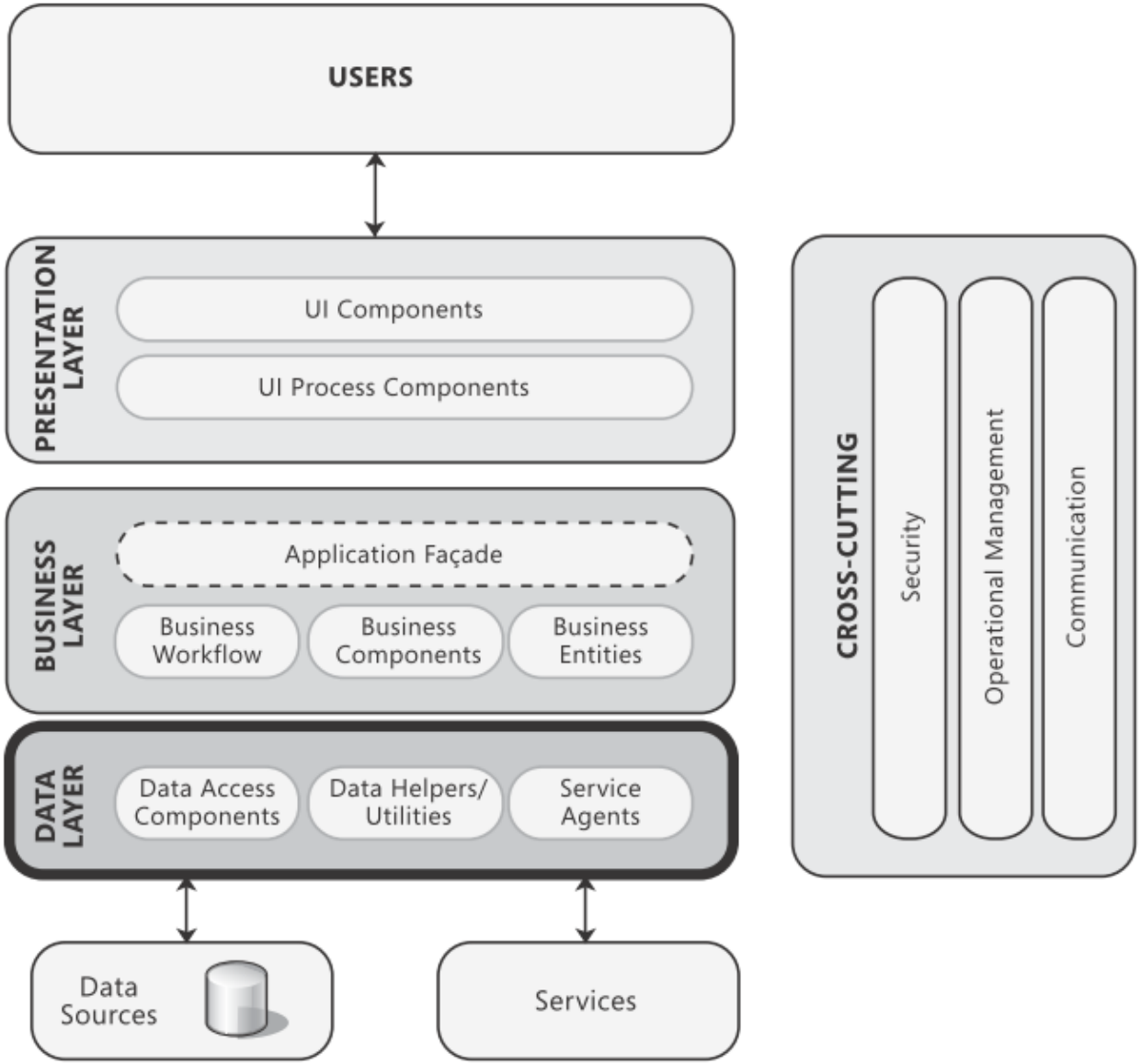
Transacções e Manutenção de Estado



Evocações longas: bloqueio de recursos muito oneroso.

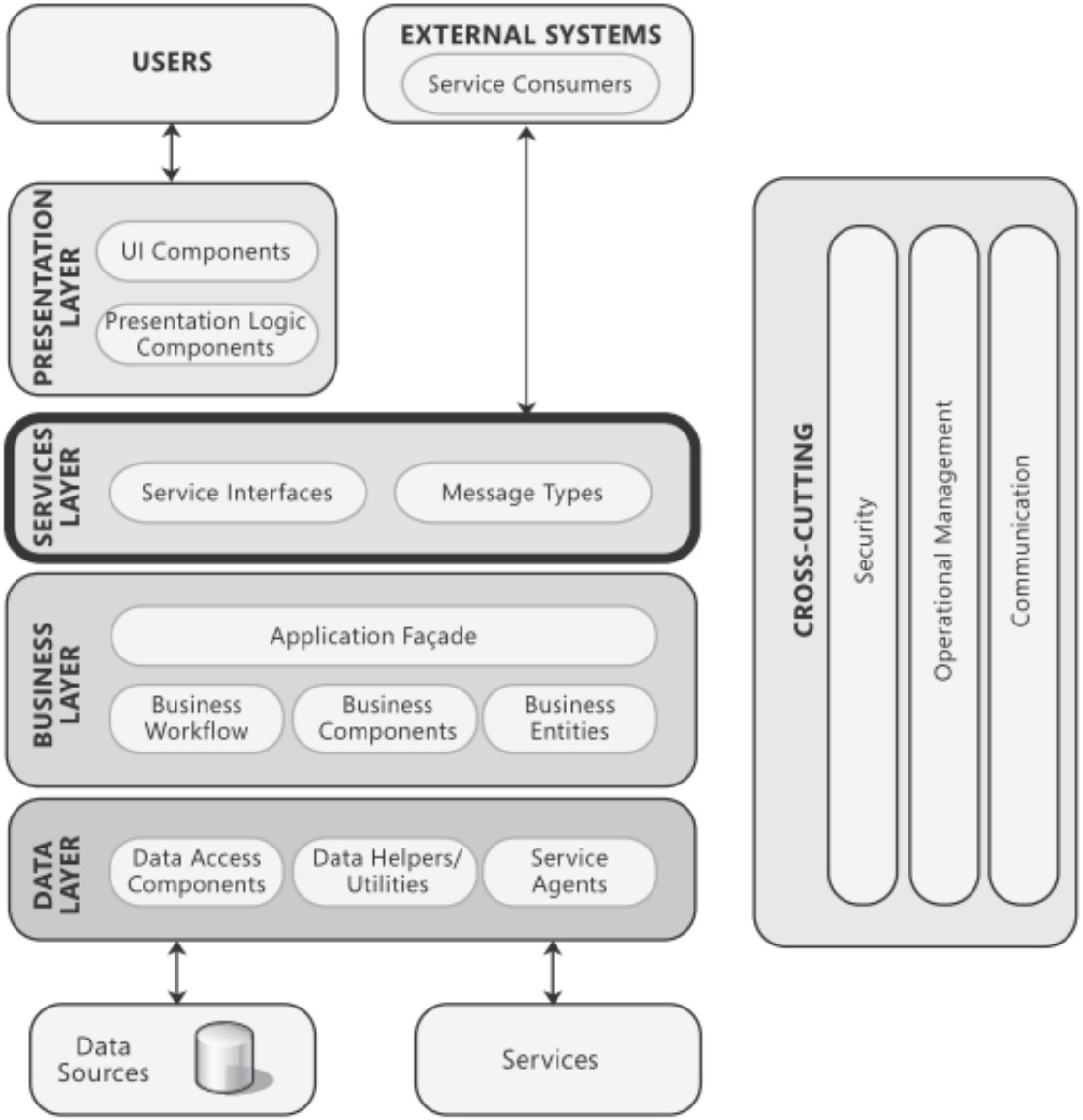
Evocações curtas: Bloqueio de recursos menos oneroso. Custo de reconstrução do estado tipicamente elevado (compensa a manutenção de estado).

Mecanismos de Acesso a Dados



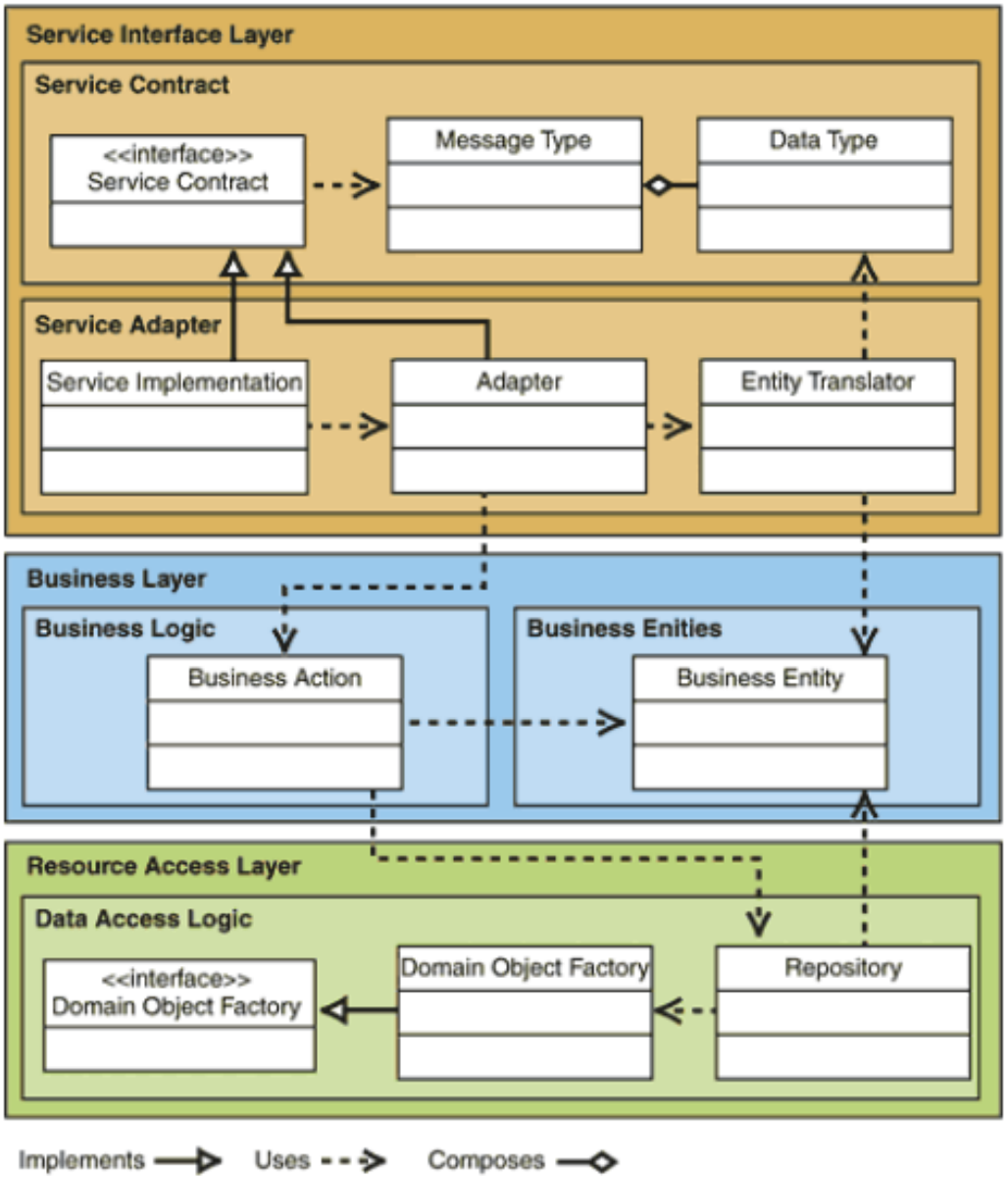
[Microsoft, 2009]

Mecanismos de Serviços



[Microsoft, 2009]

Arquitectura Multi-Camada



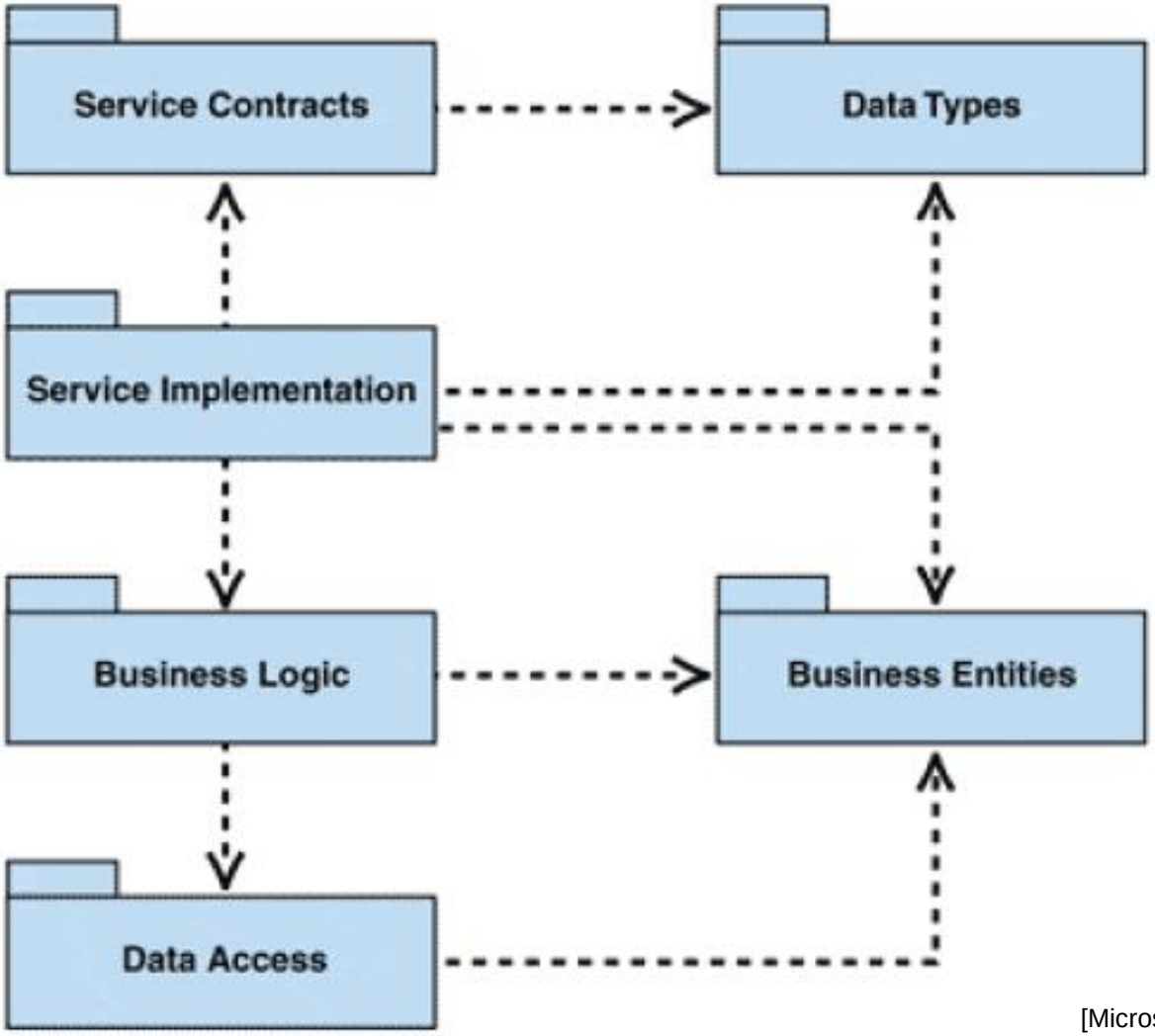
[Microsoft Developer Network]

Business Objects vs. *Entity Objects*

- ***Business Objects***
  - Objects designed primarily around the *responsibilities and behaviors* defined by a **business *use case***.
  - It is all about making it easy to build use case-derived objects that have ***business logic*, *validation rules* and *authorization rules***.
- ***Entity Objects***
  - Objects designed primarily as ***data containers***.
  - Support easy and intuitive process to *get data into and out of databases* (or other data stores) into entity objects, and then to *reshape those entity objects in memory*.

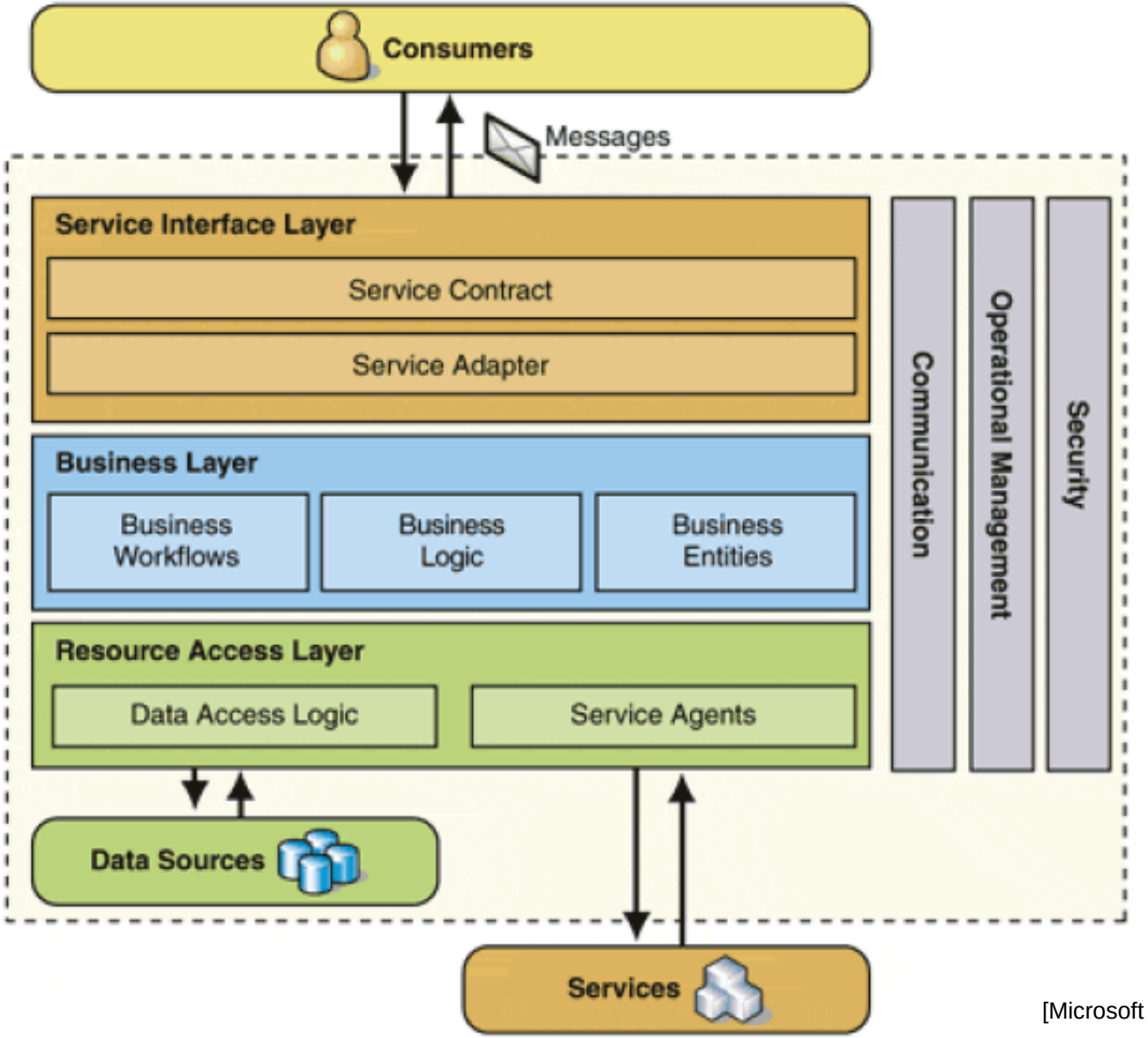
[Microsoft, 2009]

Arquitectura Multi-Camada



[Microsoft Developer Network]

Arquitectura Multi-Camada



[Microsoft Developer Network]

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.

[Microsoft, 2009]

Microsoft Application Architecture Guide: Patterns & Practices, 2nd Ed., Microsoft, 2009.

[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.

[Burbeck, 1992]

S. Chatterjee; *Messaging Patterns in Service-Oriented Architecture – Part 1*, Cap Gemini Ernst & Young - MSDN, 2004.

[Booch, 2004]

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