Análise e Projeto

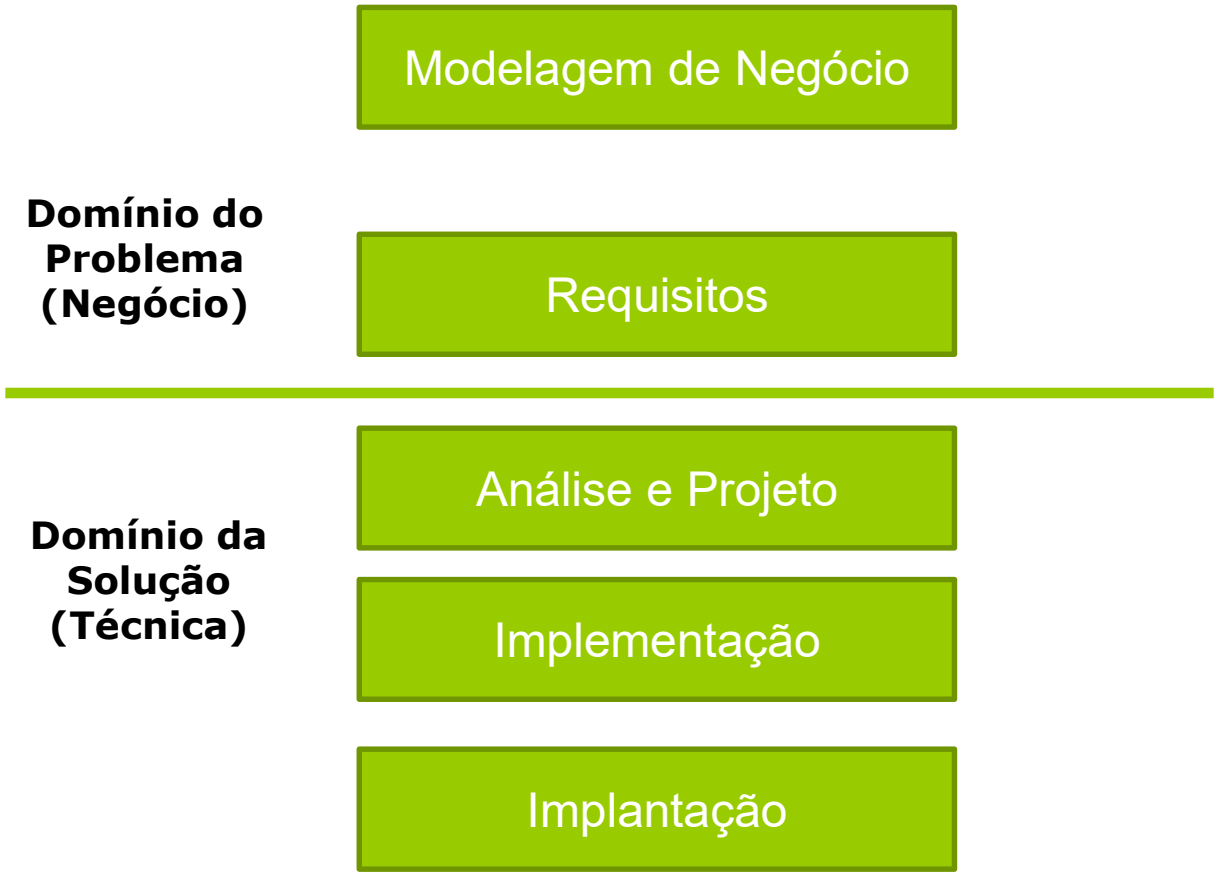
Padrões de Análise, Arquitetura e Projeto

Análise e Projeto

- O foco da disciplina de Requisitos foi descrever o comportamento de cada caso de uso
  - Para a descrição do comportamento foi utilizada linguagem natural para descrever o comportamento
  - O comportamento de cada caso de uso está no nível do negócio
  - Diagrama de Classes para descrever as classes relacionadas com o Negócio
- O foco das disciplinas de Análise e Projeto é elaborar uma "solução técnica" capaz de satisfazer os requisitos do usuário
  - Para isto é necessário conhecer as tecnologias envolvidas
  - A solução técnica compreende toda a estrutura interna no software
  - Envolve as técnicas próprias da computação
  - A solução técnica será consumida pelas pessoas responsáveis pela implementação (codificação) do software

Negócio x Solução Técnica

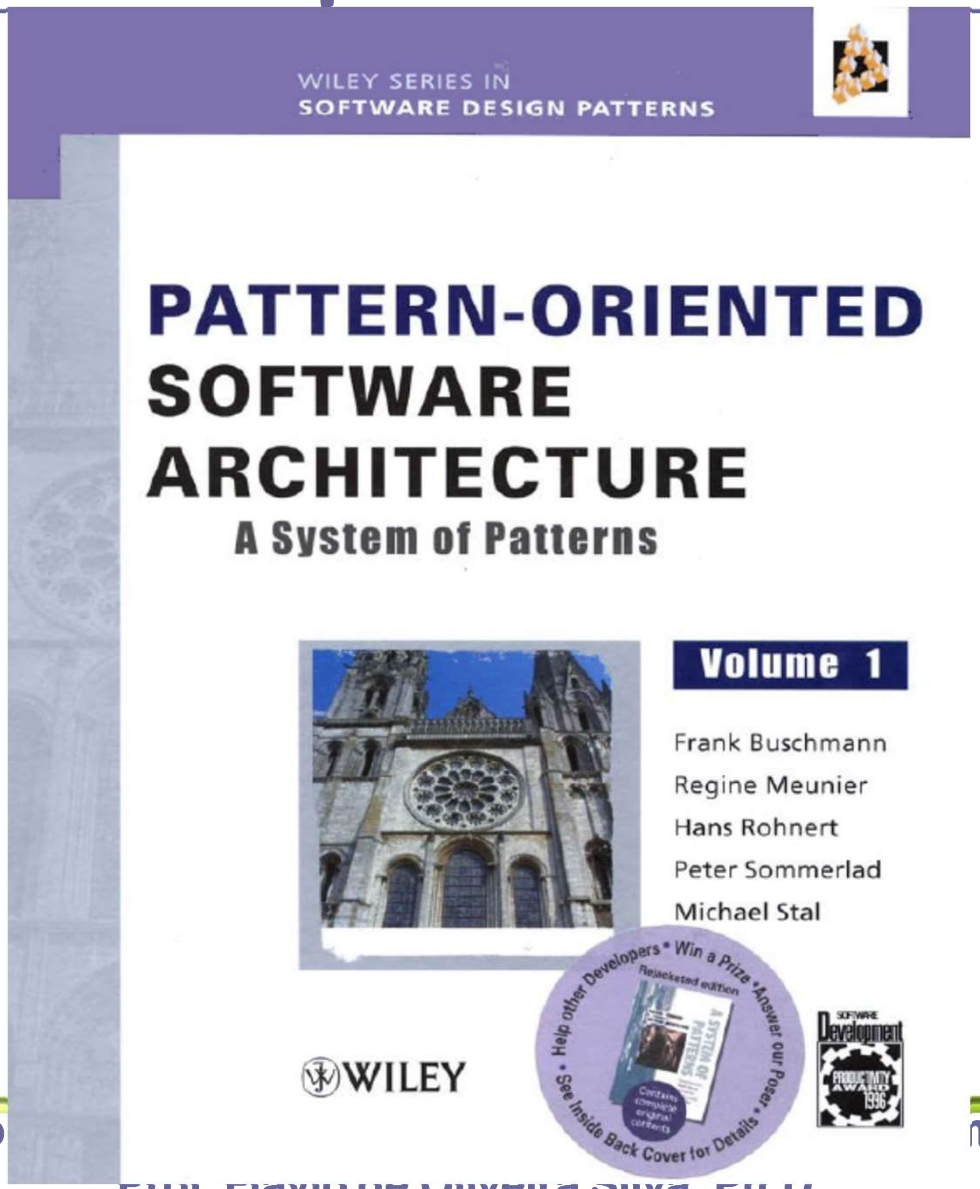
- Diferenças entre as disciplinas



Análise e Projeto
Cronograma – 2020/03

- Análise e Projeto
  - Protótipo de telas (D1) – 09/11/2020
    - Entrega parcial para comentários (D1) – 05/11/2020
  - Persistência Dados (D2) – 16/11/2020
    - Entrega parcial para comentários (D2) – 12/11/2020
  - Integrações e Protótipo da Arquitetura (D3) – 23/11/2019
    - Entrega parcial para comentários (D3) – 19/11/2020
  - Diagramas com Visões da Arquitetura (D4) – 30/11/2019
    - Entrega parcial para comentários (D4) – 26/11/2020
  - Documento de Arquitetura (D5) – 07/12/2019

Padrões de Arquitetura



Padrões

- **Nome** do padrão
- **Problema:** quando aplicar o padrão? Descreve o problema e seu contexto.
- **Solução:** elementos que definem o projeto, seus relacionamentos, responsabilidades e colaborações. É como um “template”que pode ser usado em várias situações.
- **Conseqüências** são resultados e *trade-offs* (compromissos) de se aplicar os padrões. Relacionadas aos *trade-offs* de espaço e tempo. Como o reúso é um fator importante as conseqüências incluem o impacto na flexibilidade, estensibilidade e portabilidade do sistema.

Classificação de Padrões Padrões

- Os padrões de projeto podem ser classificados de acordo com a fase de desenvolvimento em que são mais adequados:
  - Padrões de Análise (Analysis patterns)
    - Seu foco é na fase de análise ou modelamento de negócio
    - Padrões ligados ao domínio do problema
  - Padrões de Arquitetura (Architectural patterns)
    - Seu foco é na arquitetura do software
  - Padrões de Projeto (Design patterns)
    - Foco no projeto de componentes do software
- Muitas das vezes os padrões podem estar muito ligados tanto ao domínio da solução, quanto do problema

Classificação de Padrões Padrões

- Padrões de Análise (Analysis patterns)
  - Martin Fowler , 1996
- Padrões de Arquitetura (Architectural patterns)
  - Apresentado inicialmente por Frank Buschmann et al., 1996
  - Computação Distribuída - Frank Buschmann et al., 2007
- Padrões de Projeto (Design patterns)
  - GOF (Gang of Four) E. Gamma, R. Helm, R. Johnson, J. Vlissides – 1995
  - Aplicações Concorrentes e em Rede - Frank Buschmann et al. – 2000
  - Enterprise Integration Patterns – Gregor Hohpe, 2003
  - Real-time Design Patterns – Bruce Douglass, 2003
  - .Net Design Patterns - Christian Thilmany, 2003
  - J2EE Design Patterns - Deepak Alur, 2003
  - Web Services Patterns – Paul Monday, 2003
  - Ajax Design Patterns - Michael Mahemoff, 2006
  - SOA Design Patterns – Thomas Erl, 2009

Padrões de Análise (Analysis Patterns)

- Proposto por Martin Fowler, em livro publicado em 1996
- Notação do Livro não é baseada em UML
- Baseada em áreas (domínios) específicas como: manufatura; financeira e saúde
- Mesmo assim, padrões podem apresentados podem ser úteis em outros domínios
- Alguns princípios apresentados
  - Um modelo não está certo ou errado, eles podem ser mais ou menos úteis
  - Modelos conceituais estão ligados a tipos (interfaces) e não implementações (classes)
  - Padrões são o ponto de partida, não o destino
  - Sempre que possível, quando existir um tipo e um supertipo, considere colocar os recursos no supertipo, desde que isto faça sentido
  - Quando múltipos atributos possuam um comportamento relacionado e presente em muitos tipos, combine estes atributos em um novo tipo fundamental

Padrões de Análise (Analysis Patterns)

- Exemplos de alguns padrões de projeto
  - Quantity (3.1)
  - Conversion Ratio (3.2)
  - Compound Units (3.3)
  - Measurement (3.4)
  - Observation (3.5)
  - Range (4.3)
  - Name (5.1)
  - Account (6.1)
  - Transaction (6.2)
  - Summary Account (6.3)
  - Plan (8.4)
  - Contract (9.1)
  - Product (10.3)
  - Associative Type (15.1)

Padrões de Arquitetura (Architectural patterns)

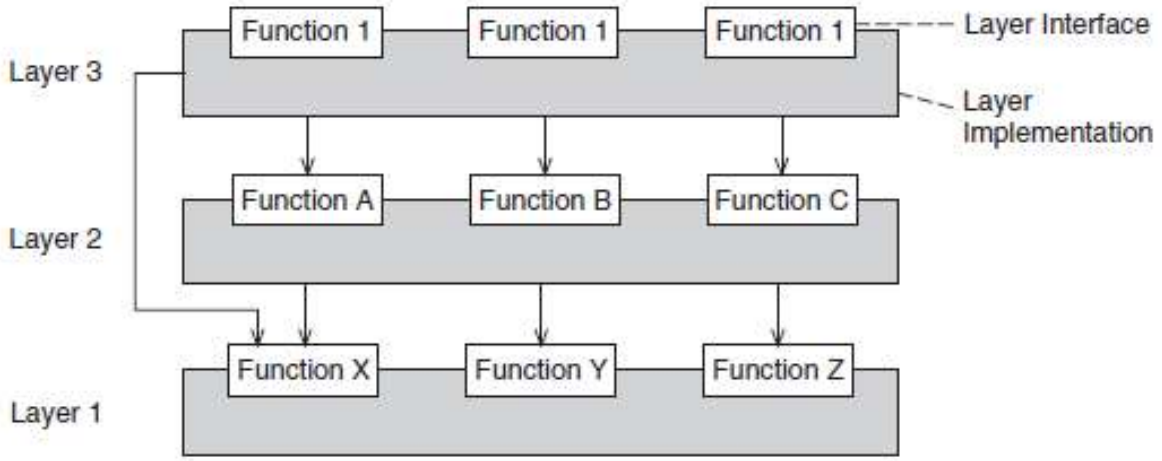
- Expressam uma organização fundamental de estrutura de um software
- O ponto de partida da análise orientada a objetos é a criação do chamado Domain Model
- O “Domain Model” expressa as classes ligadas ao domínio do problema
- Os padrões de arquitetura auxiliam na concepção do software e na transição para o domínio da solução
- Alguns padrões de arquitetura
  - Layers
  - Model-View-Controler (MVC)
  - Pipes and Filters
  - Microkernel
  - Shared Repository

Padrões de Projeto (Design Patterns)

- Trabalho proposto inicialmente por Erich Gamma, Richard Helm, Ralph Jonhson, John Vlissides (Gang of Four) em 1995
- Famílias de Padrões
  - De Criação
    - Responsáveis pela criação de objetos
    - Permitem que o sistema fique independente da forma como os objetos são criados
    - Factory, Abstract Factory, Singleton, Builder, Prototype
  - Estruturais
    - Relacionados com a forma com que classes e objetos são compostos a fim de formar estruturas maiores
    - Adapter, Bridge, Composite, Decorator, Facade, Proxy
  - Comportamentais
    - Relacionados com a atribuição de responsabilidades entre objetos
    - Descrevem a comunicação entre objetos
    - Chain of Responsibility, Command, Flyweigth, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template, Visitor

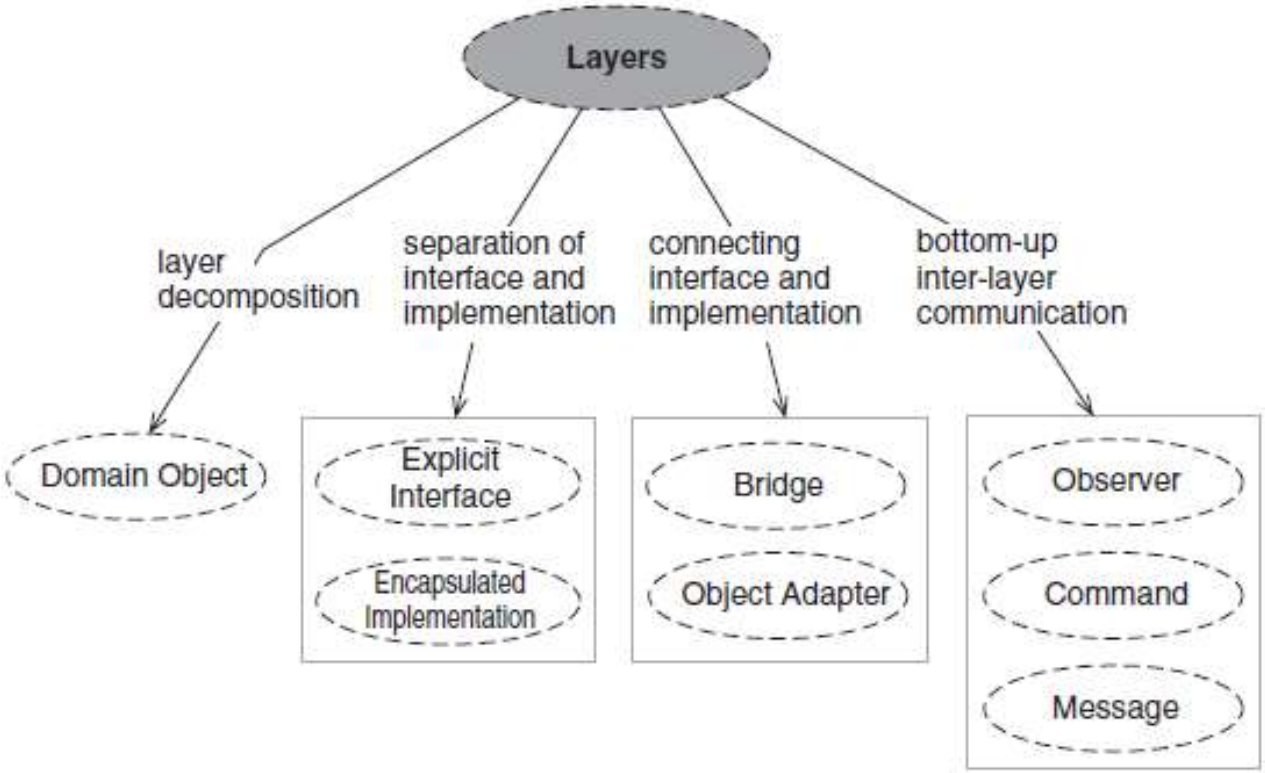
Layer

- Permite o desenvolvimento de forma independente de diferentes partes do sistema de software

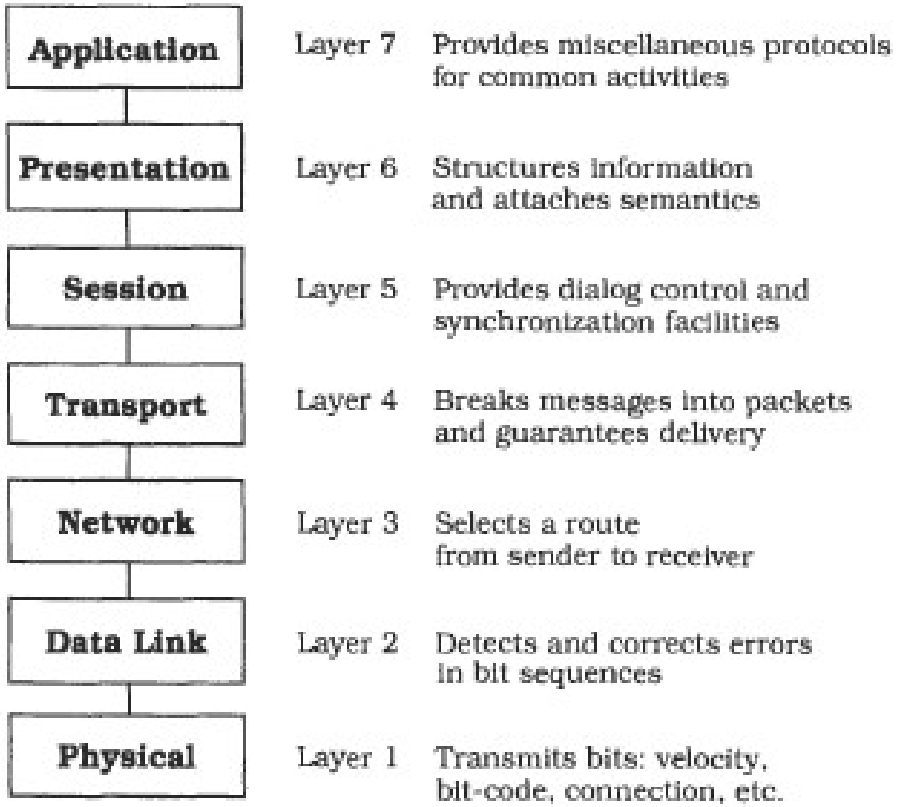


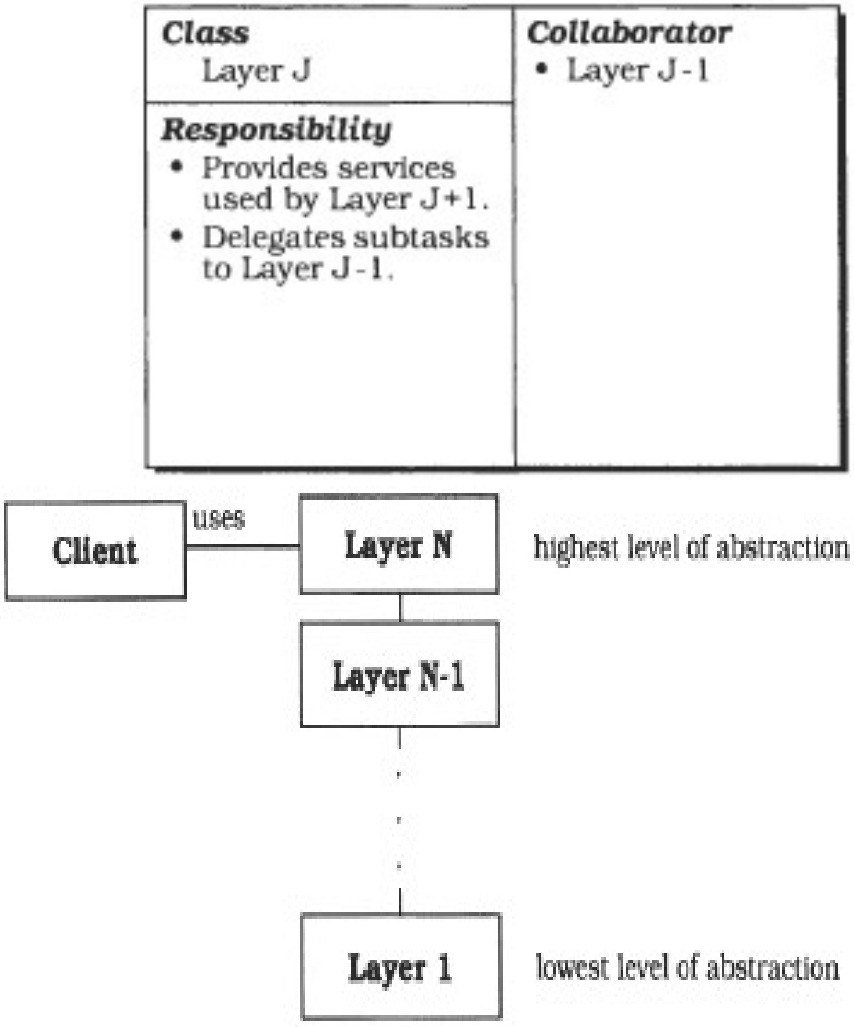
Layer

- Usos com outros padrões



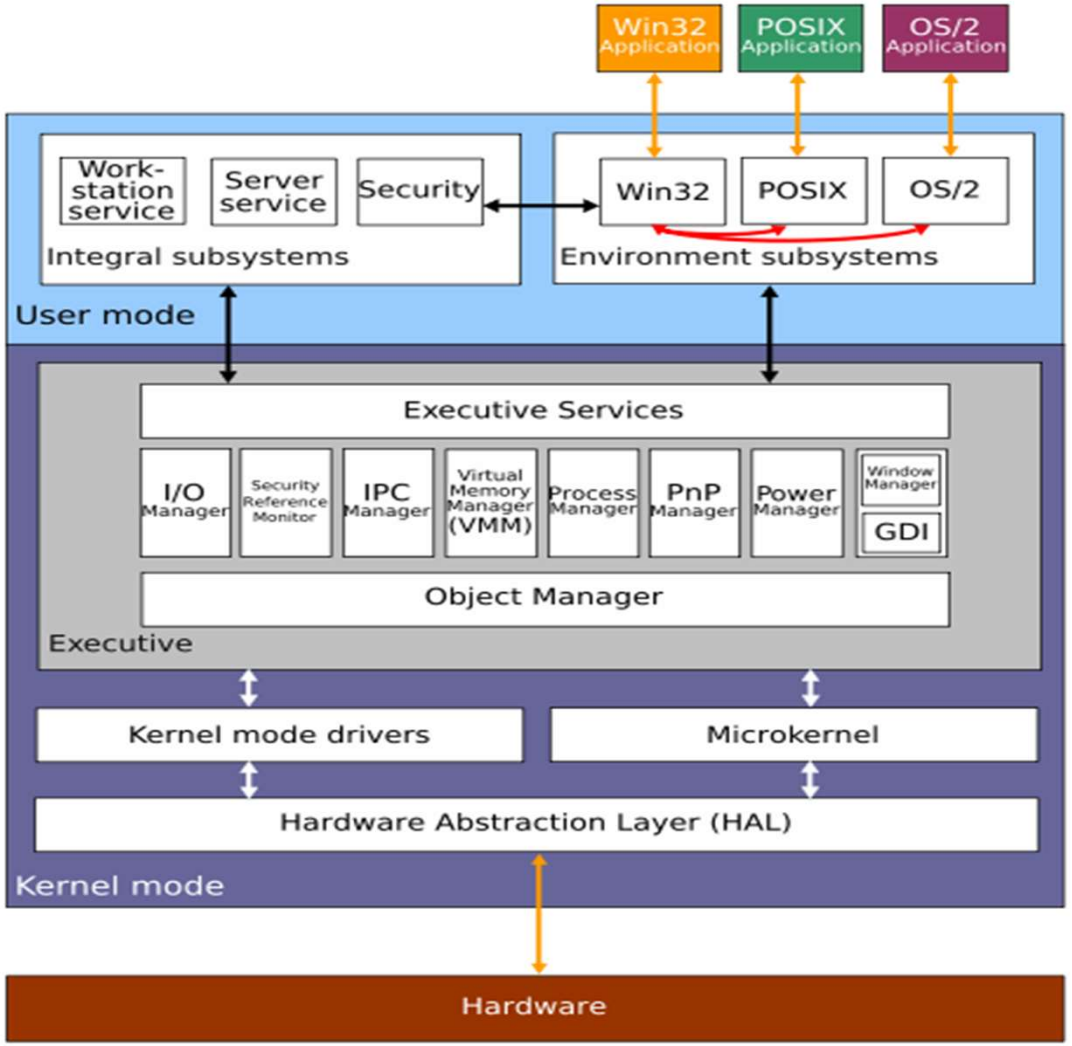
Estilos (padrões) de Arquitetura
Camadas





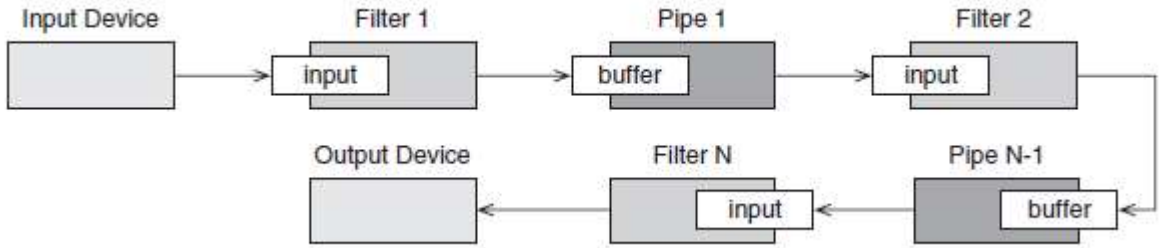
Fonte: Buschmann

Camadas (Windows NT)



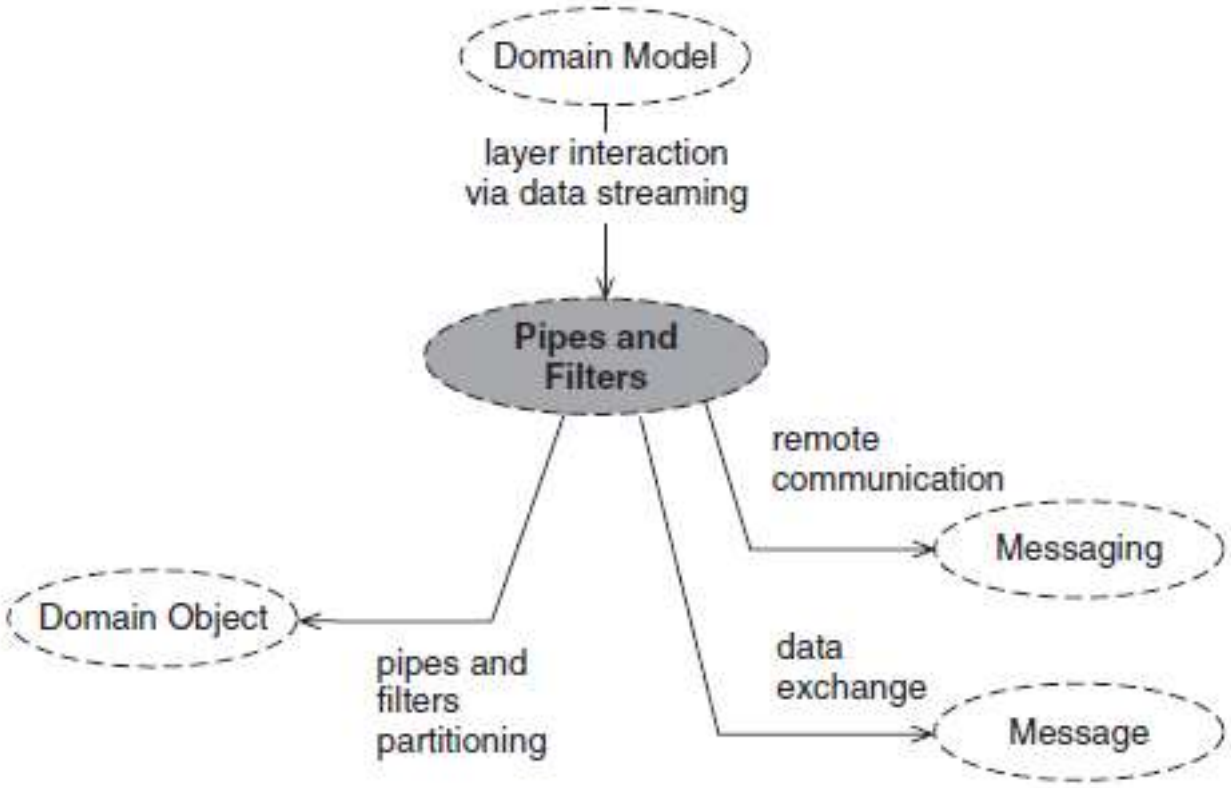
Pipes and Filters

- Arquitetura especializada em tratar fluxos de dados (data streams)

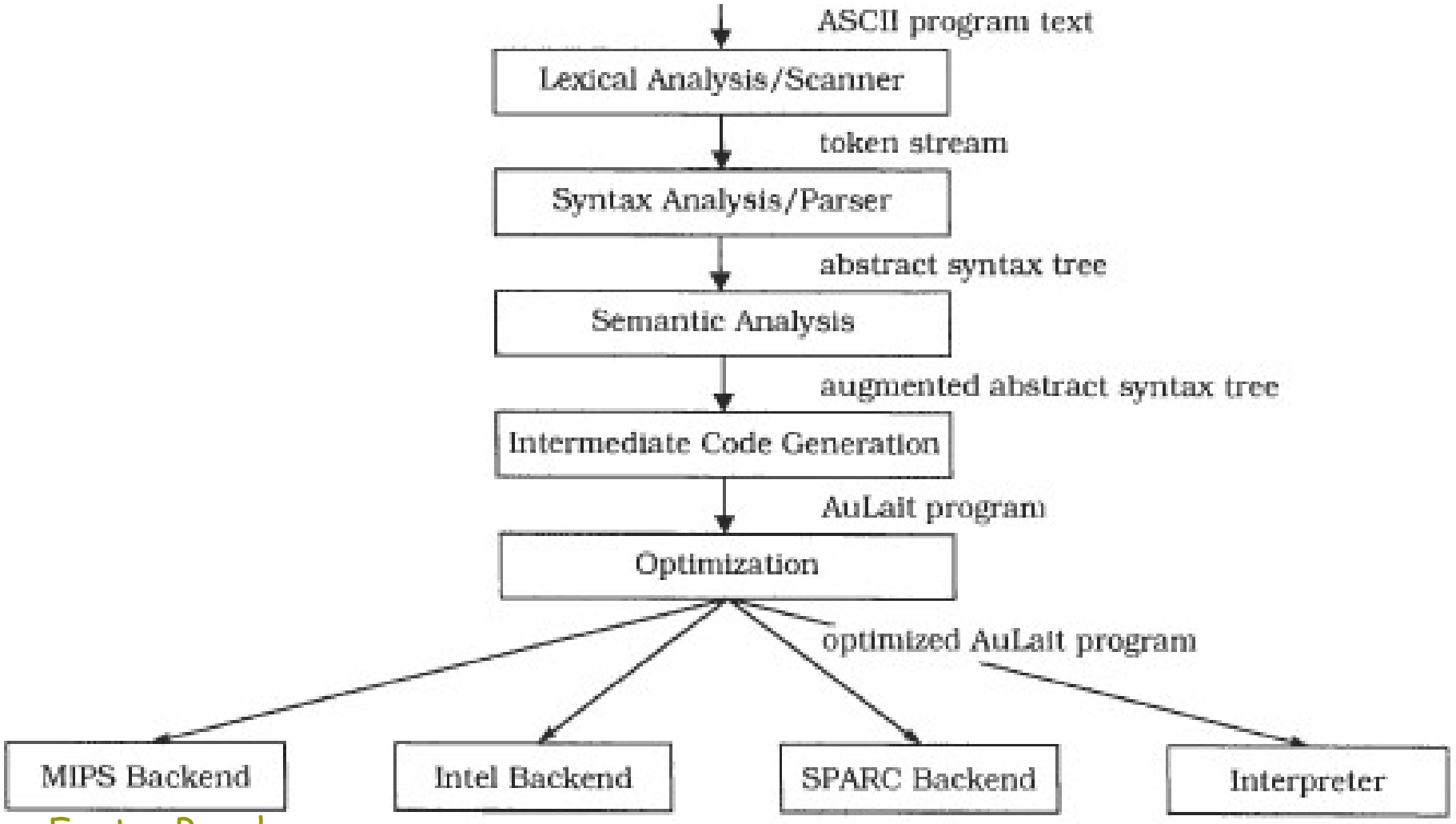


Pipes and Filters

- Uso com outros padrões



Estilos (padrões) de Arquitetura
Dutos e Filtros



Fonte: Buschmann

Estilos (padrões) de Arquitetura
Dutos e Filtros

Class Filter	**Collaborators**
Responsibility • Gets input data. • Performs a function on its input data. • Supplies output data.	• Pipe

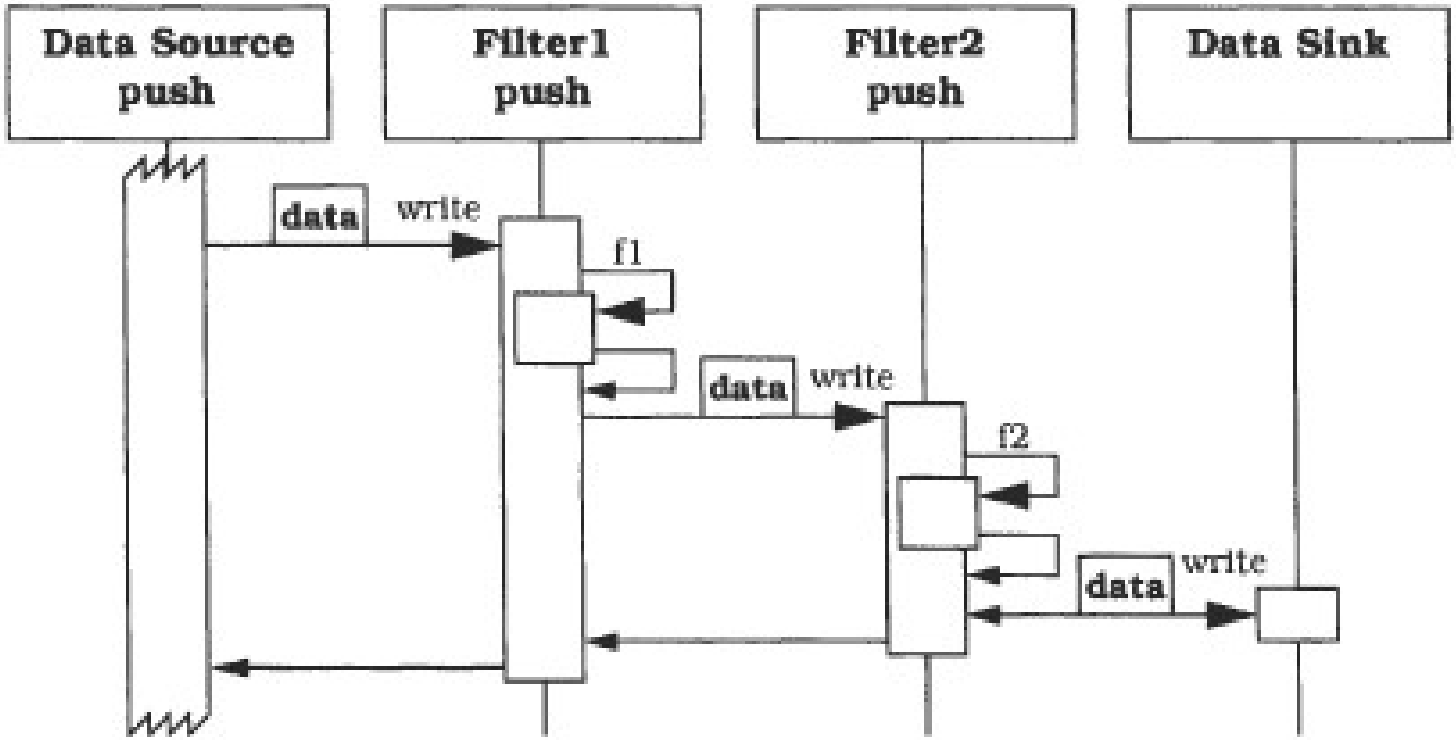
Class Pipe	**Collaborators**
Responsibility • Transfers data. • Buffers data. • Synchronizes active neighbors.	• Data Source • Data Sink • Filter

Class Data Source	**Collaborators**
Responsibility • Delivers input to processing pipeline.	• Pipe

Class Data Sink	**Collaborators**
Responsibility • Consumes output.	• Pipe

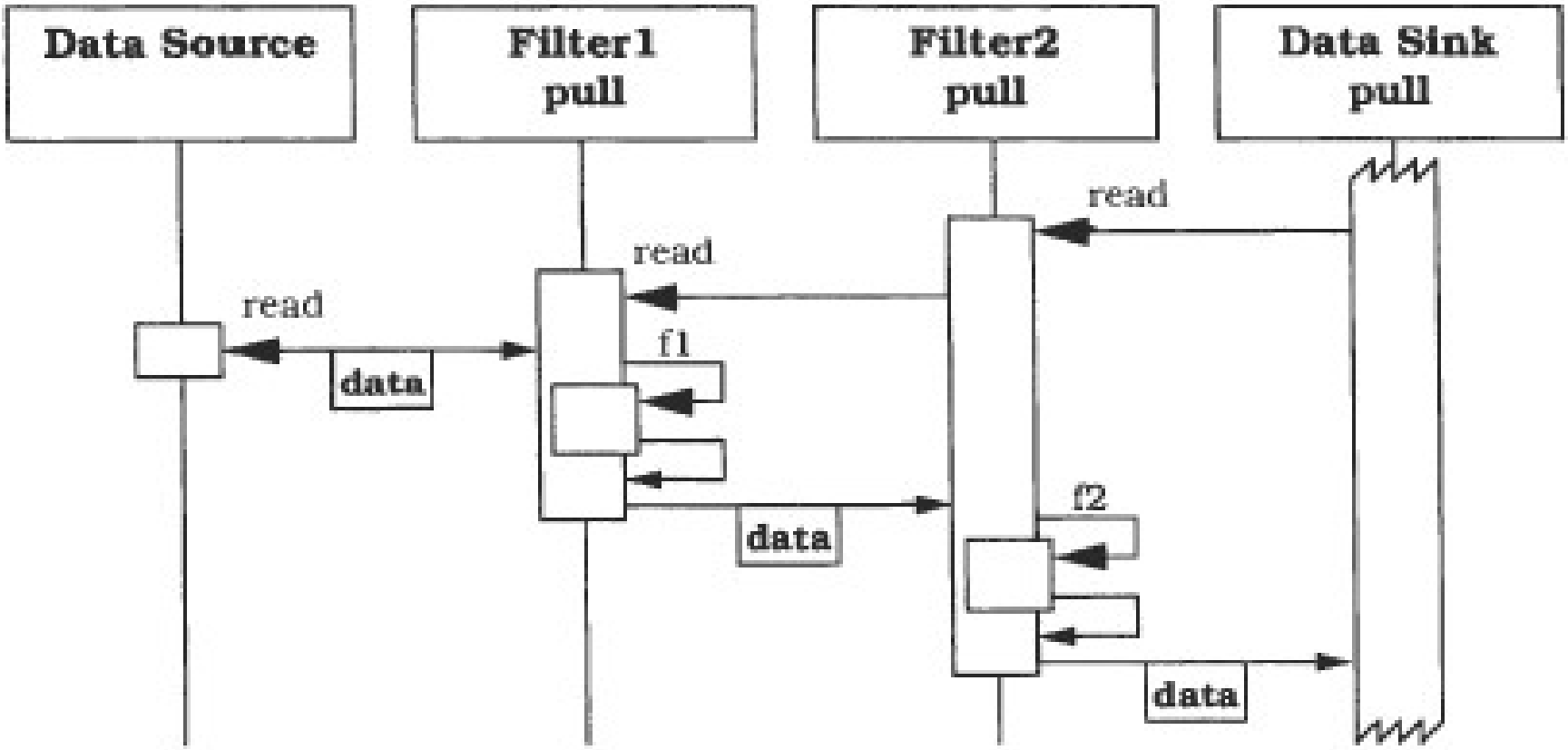
Fonte: Buschmann

Dutos e Filtros
Atividade começando com o data source



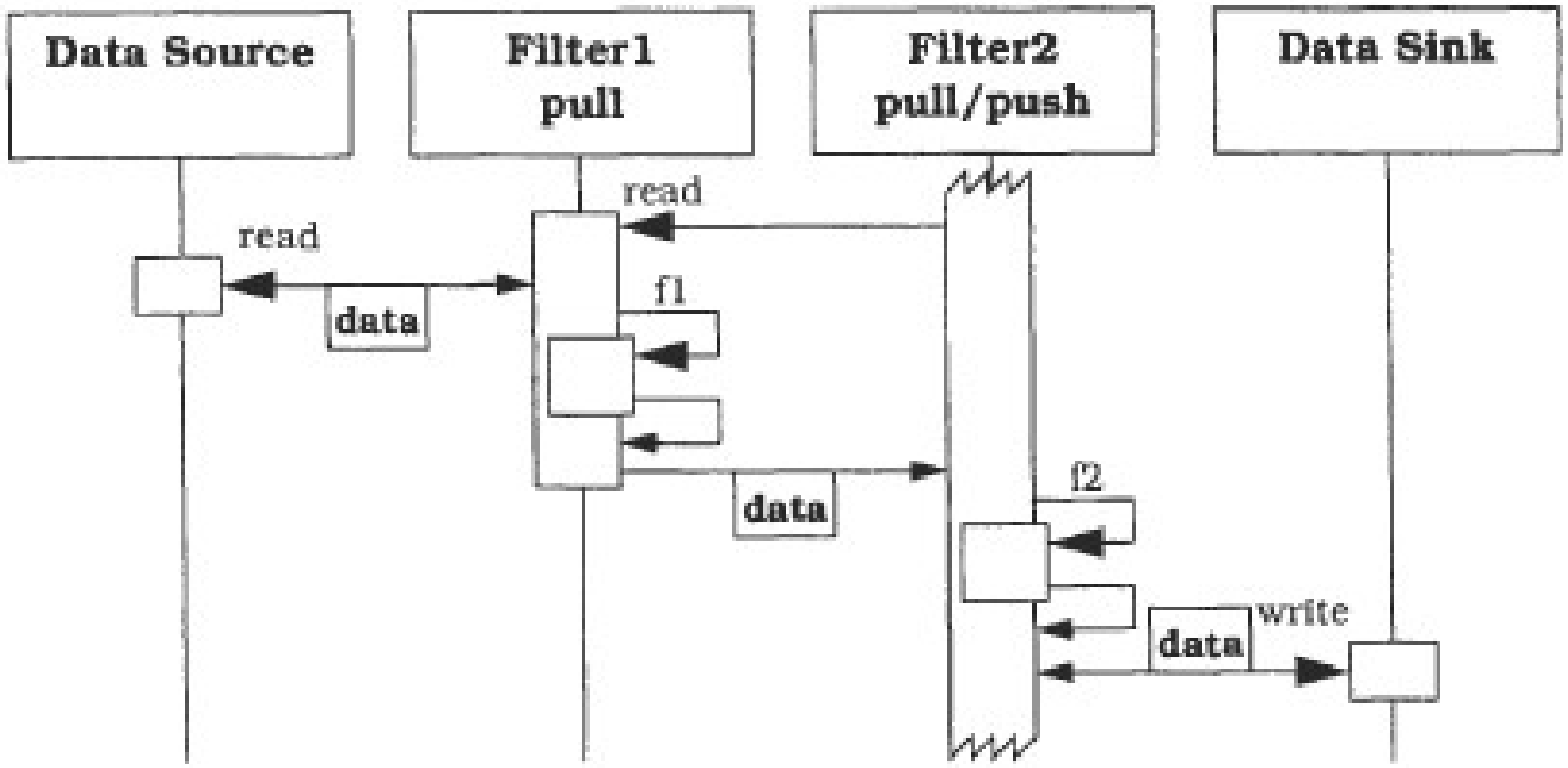
Fonte: Buschmann

Dutos e Filtros
Atividade começando com o data sink



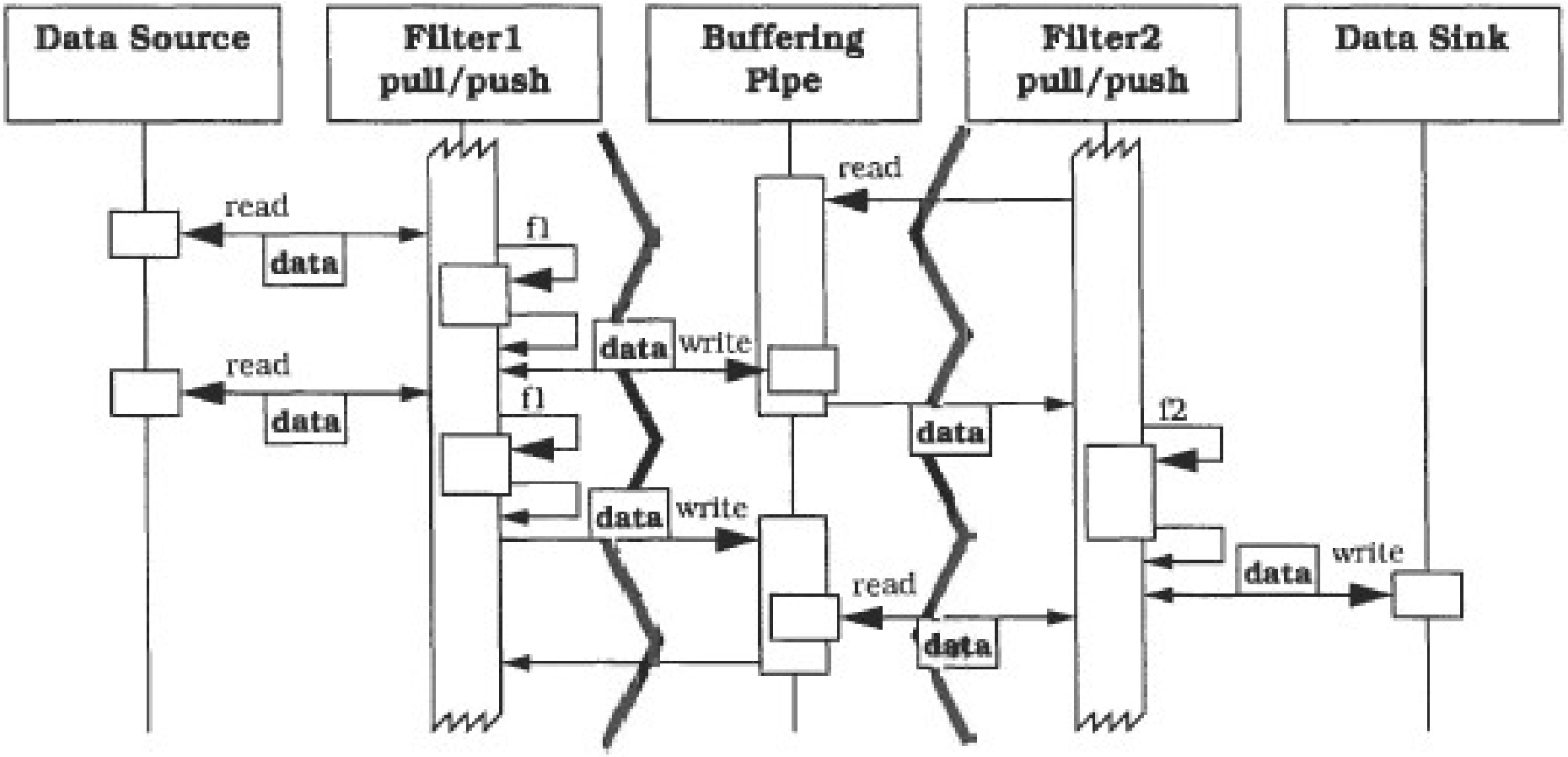
Fonte: Buschmann

Dutos e Filtros
Mix pull-push (Source/Sink passivos)



Fonte: Buschmann

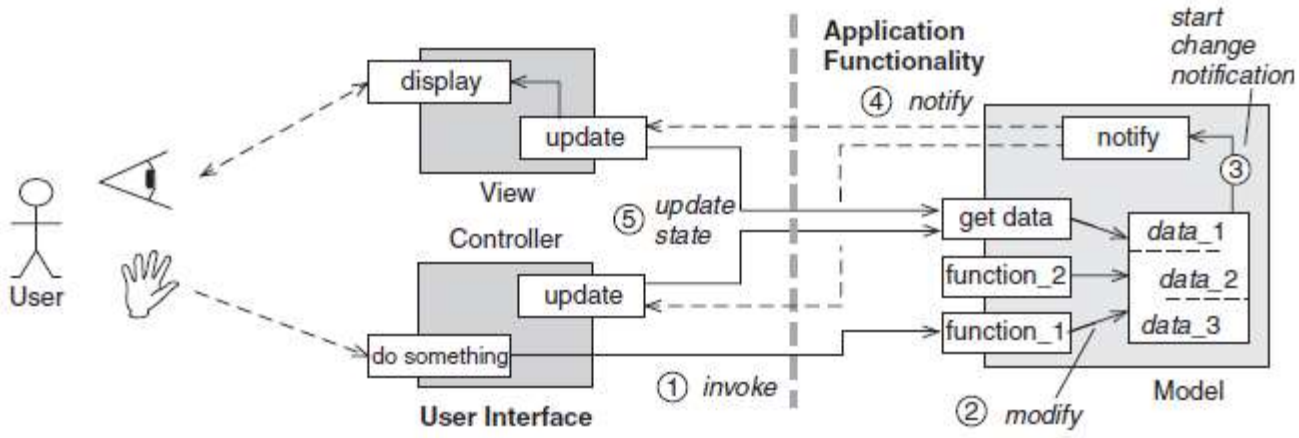
Dutos e Filtros
Filtros ativos



Fonte: Buschmann

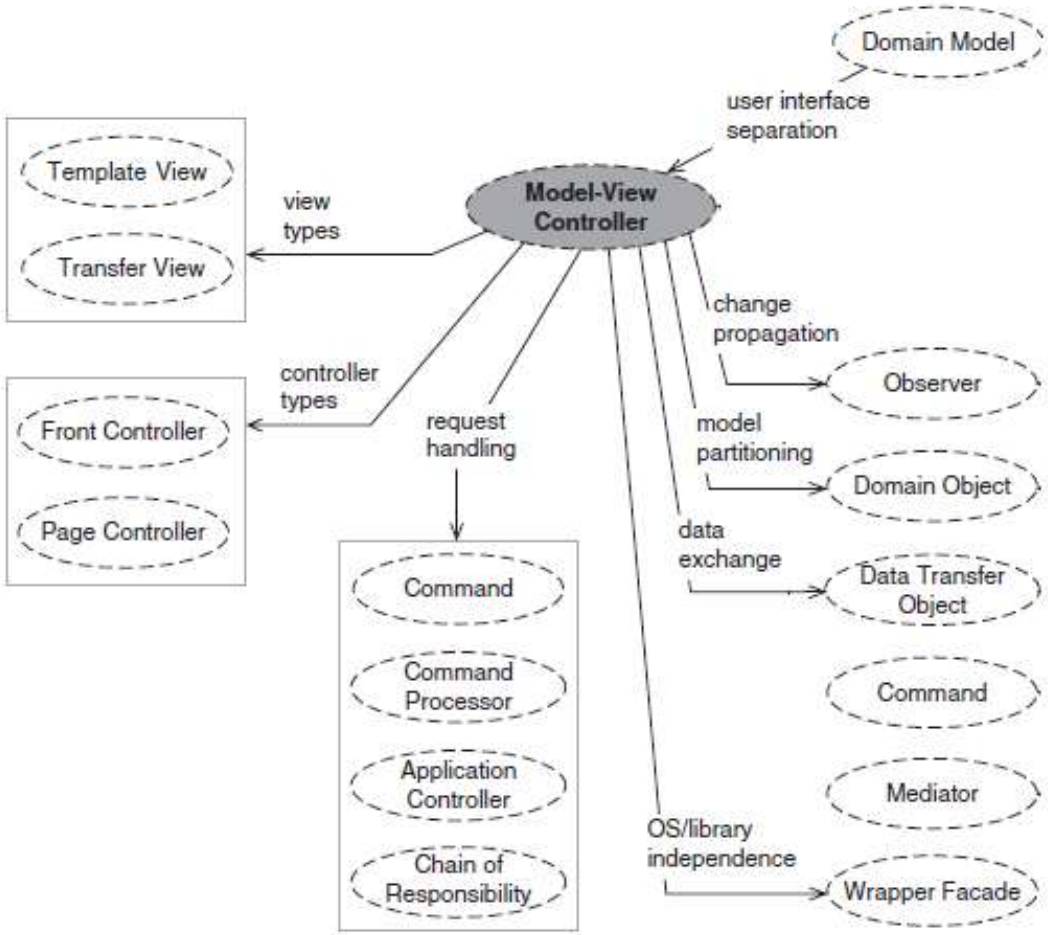
Model-View-Controller

- Uso
  - Em situações onde a interface do usuário de uma aplicação pode mudar de forma mais frequente que o seu domínio

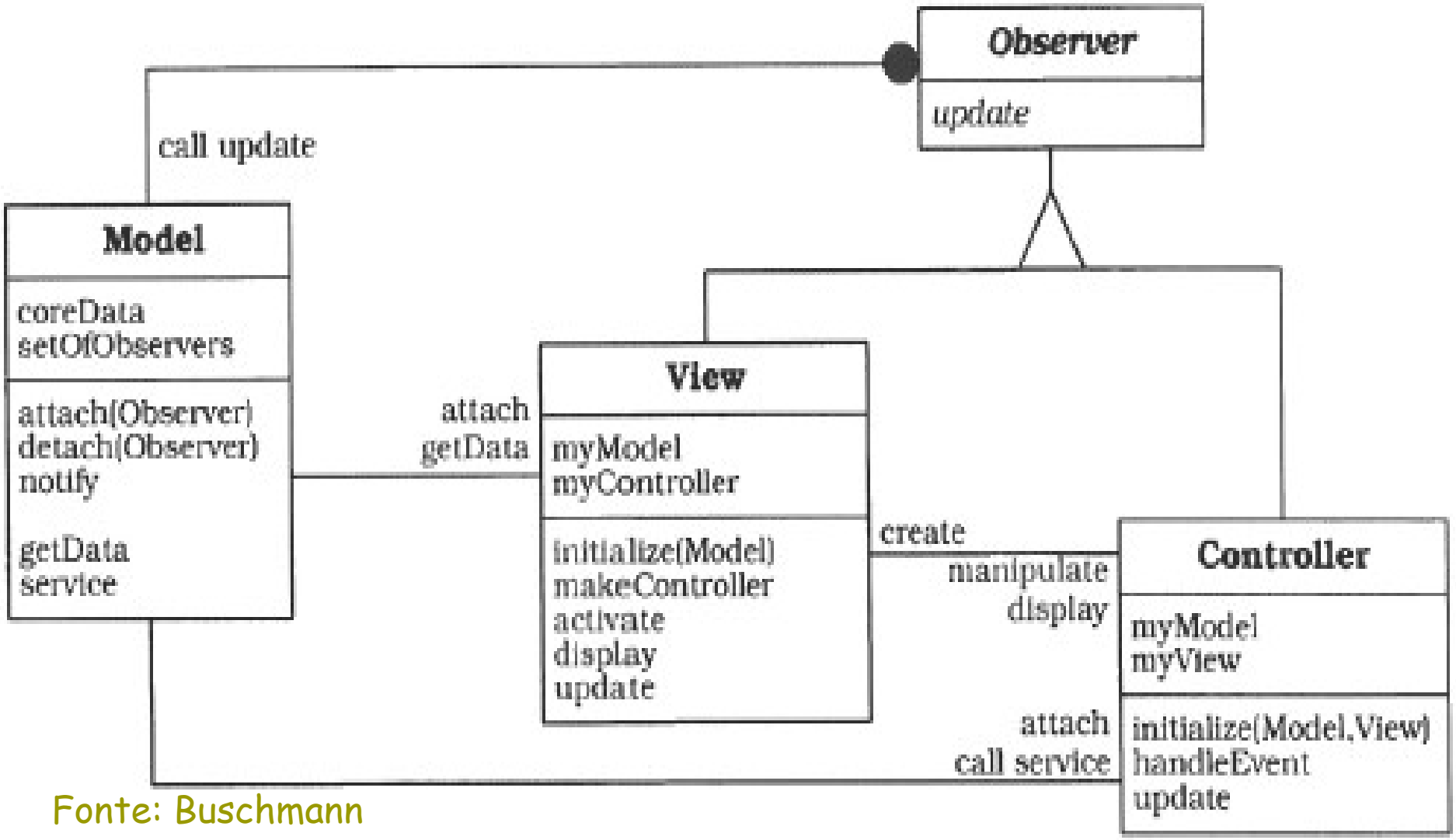


Model-View-Controller

- Uso do MVC com outros padrões

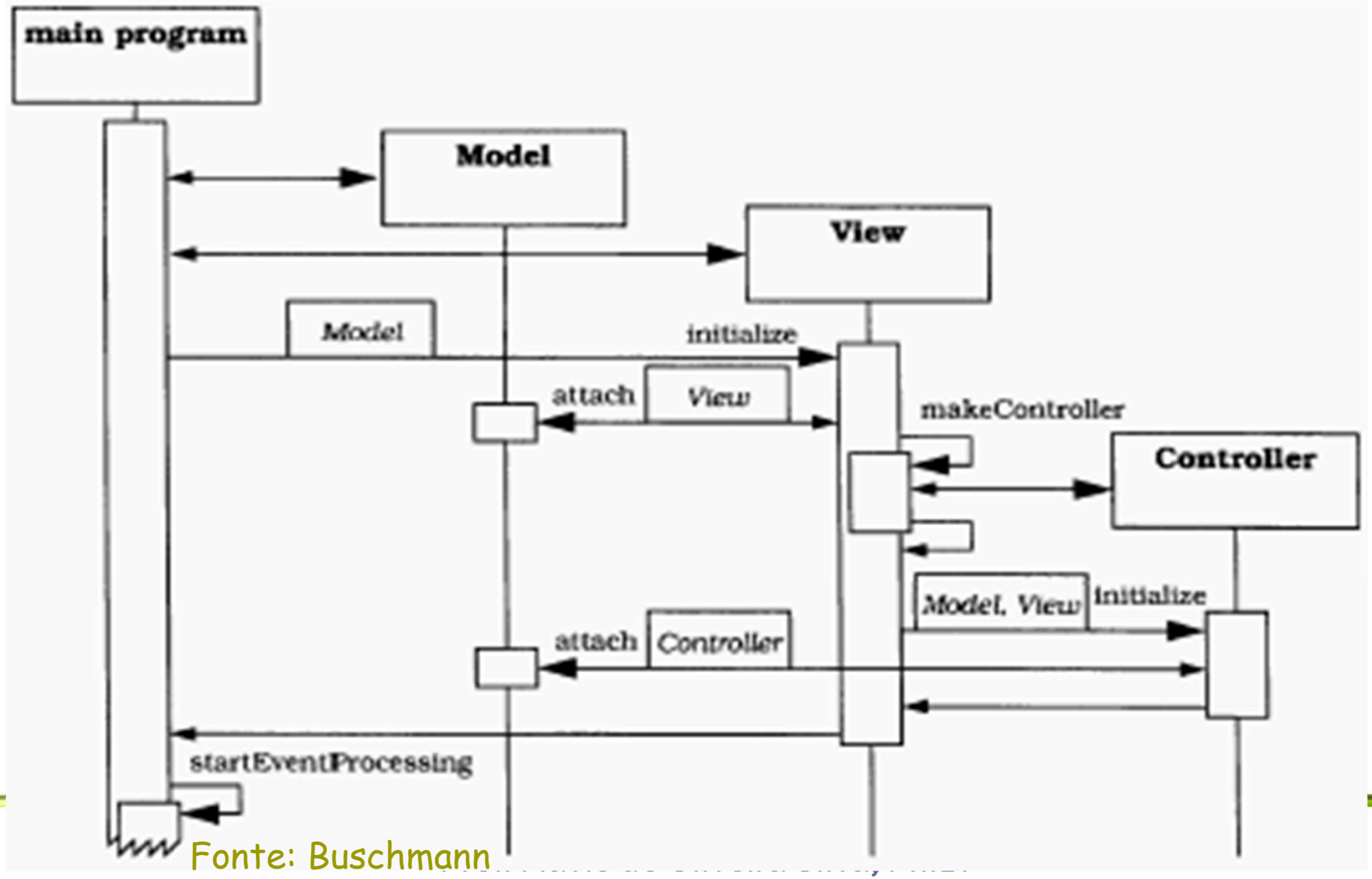


MVC - Estrutura



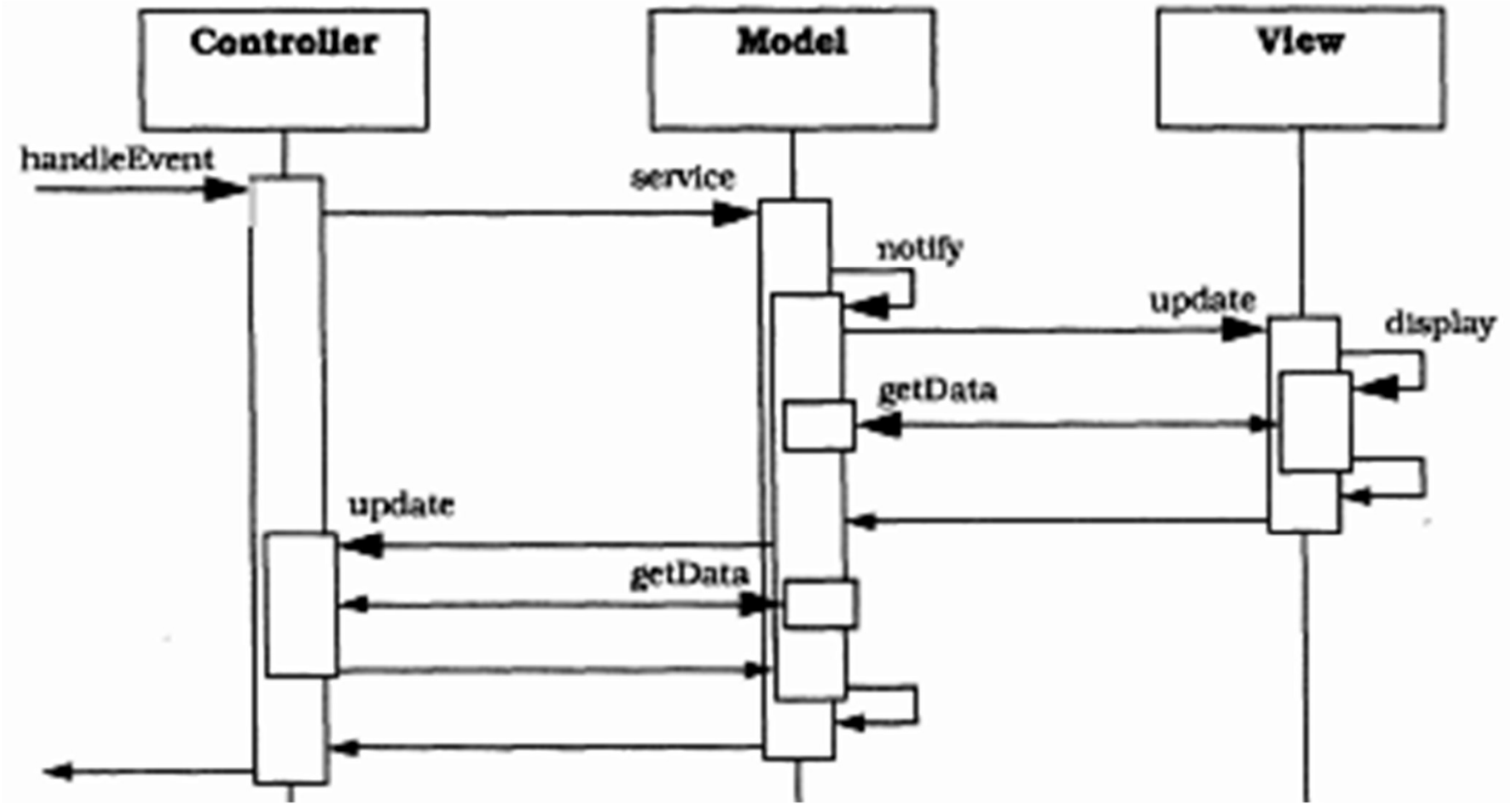
Fonte: Buschmann

MVC – Dinâmica –Inicialização



Fonte: Buschmann

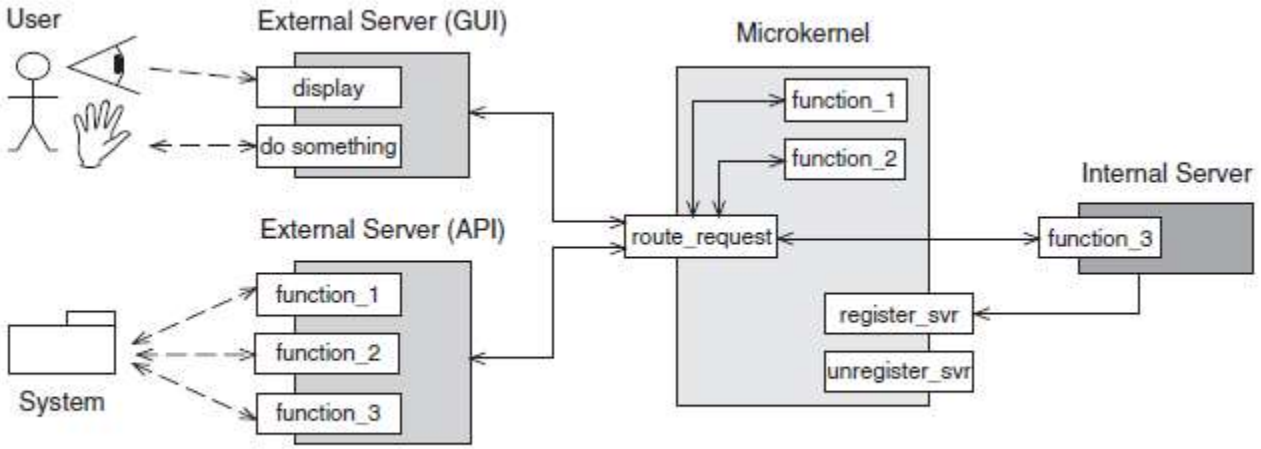
MVC – Dinâmica
Tratamento de eventos



Fonte: Buschmann

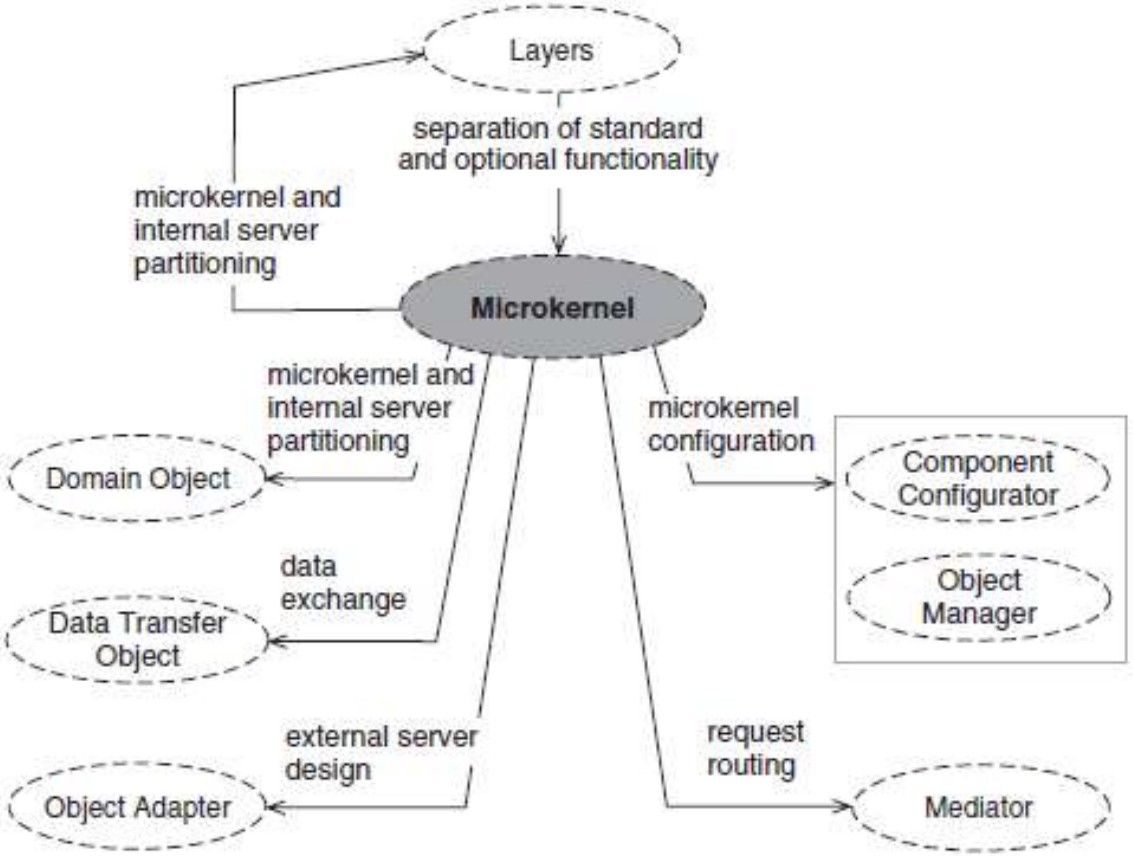
Microkernel

- Uso
  - Arquitetura com suporte para escalabilidade funcional e adaptável para diferentes cenários de implantação

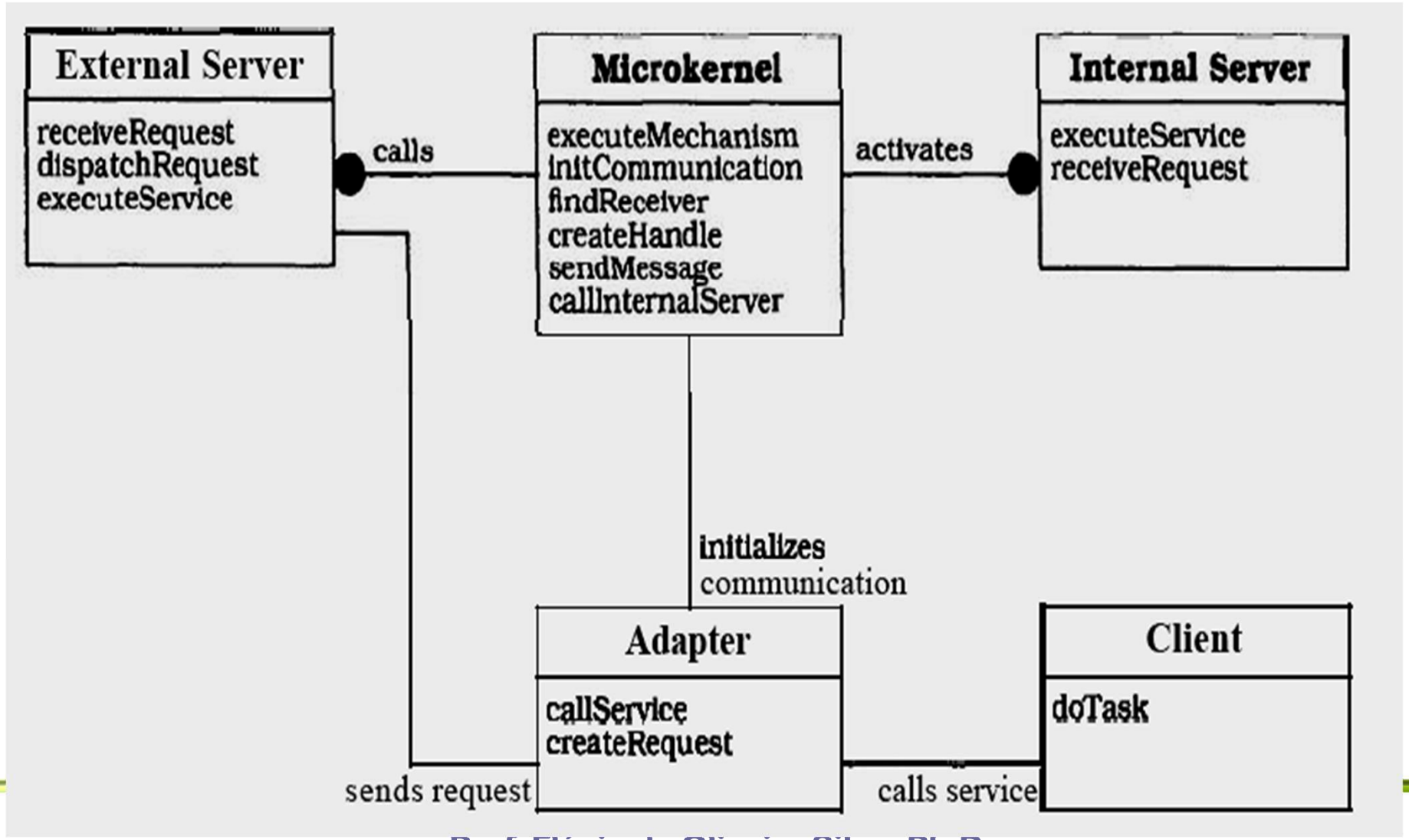


Microkernel

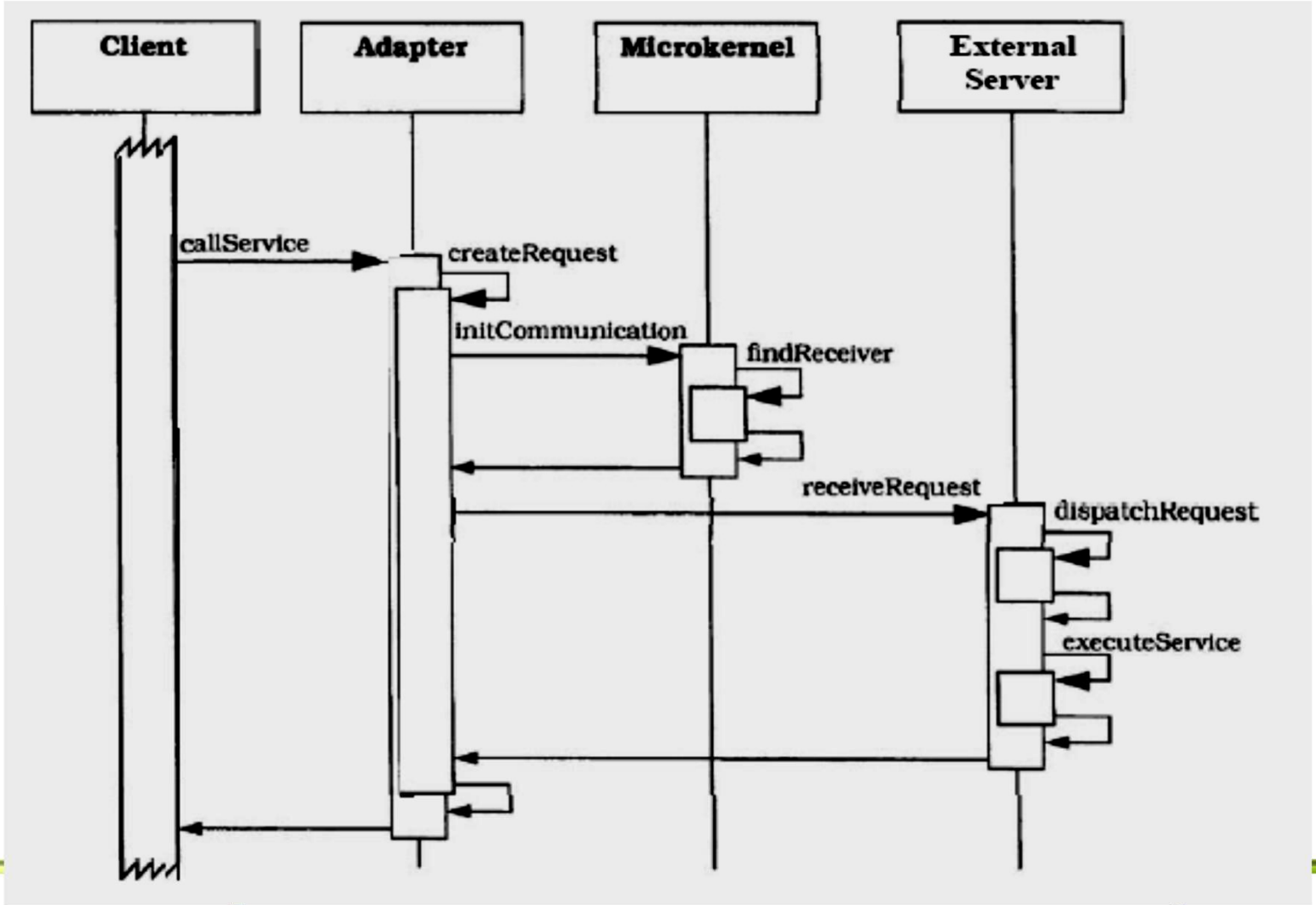
- Uso com outros padrões



Microkernel

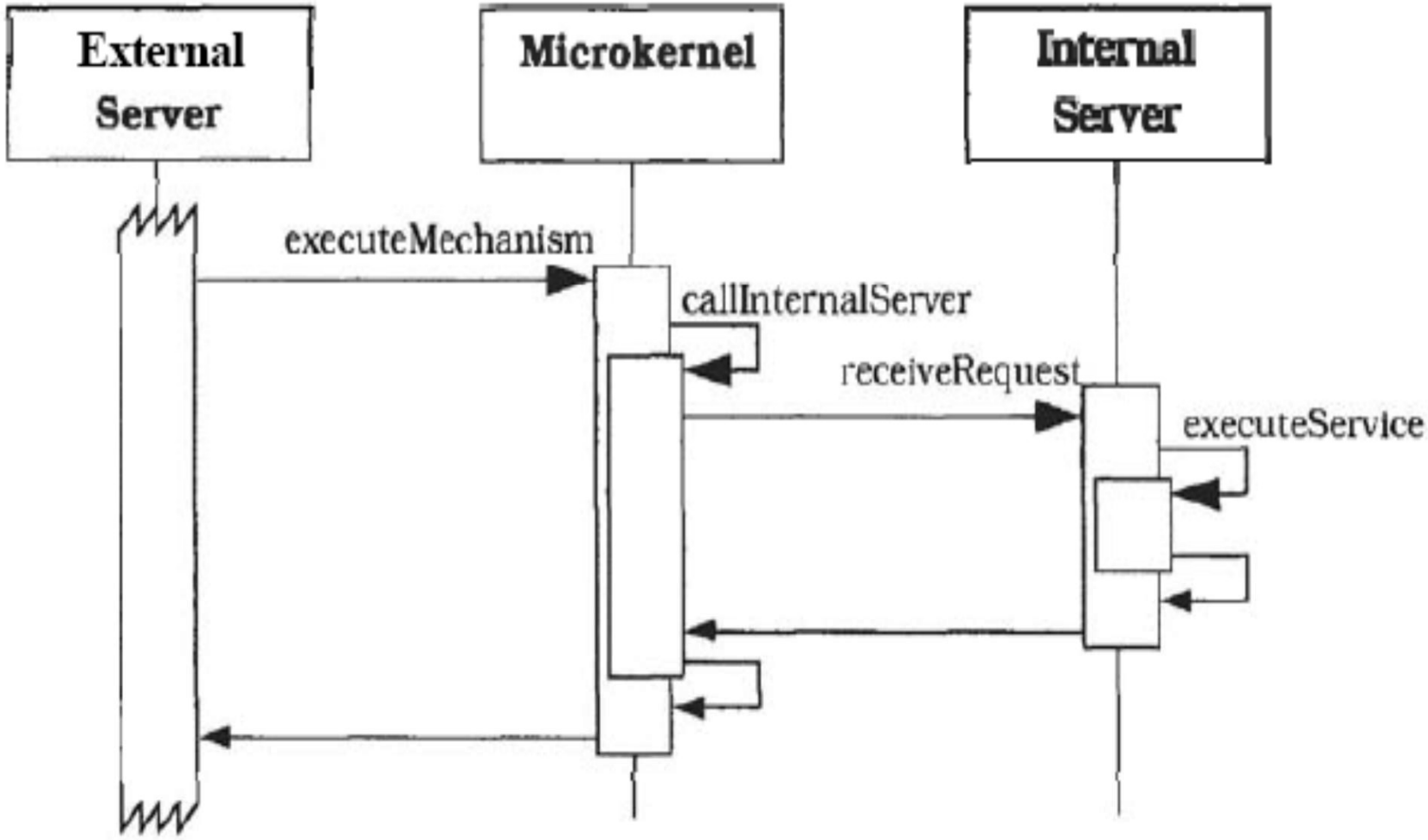


Microkernel
Cliente chamando serviço

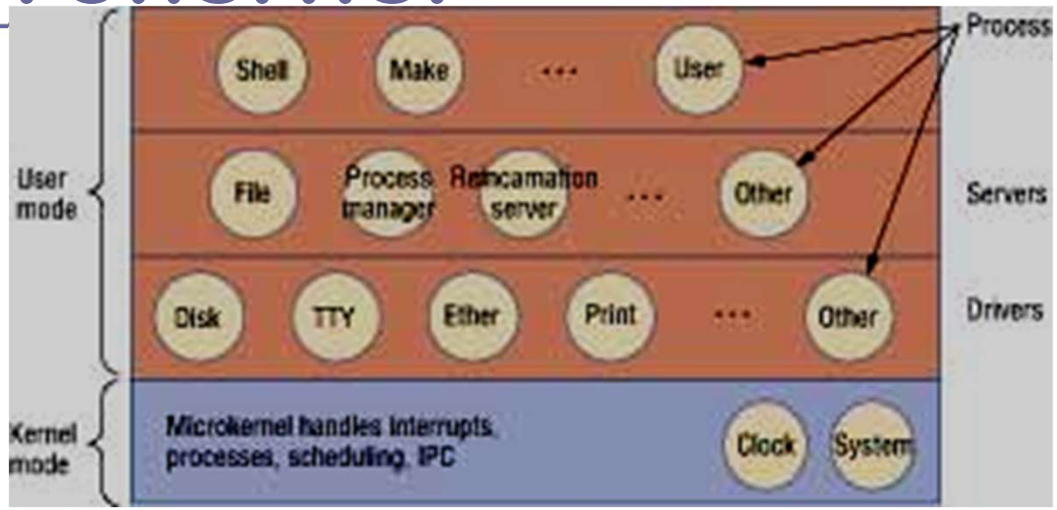


Microkernel

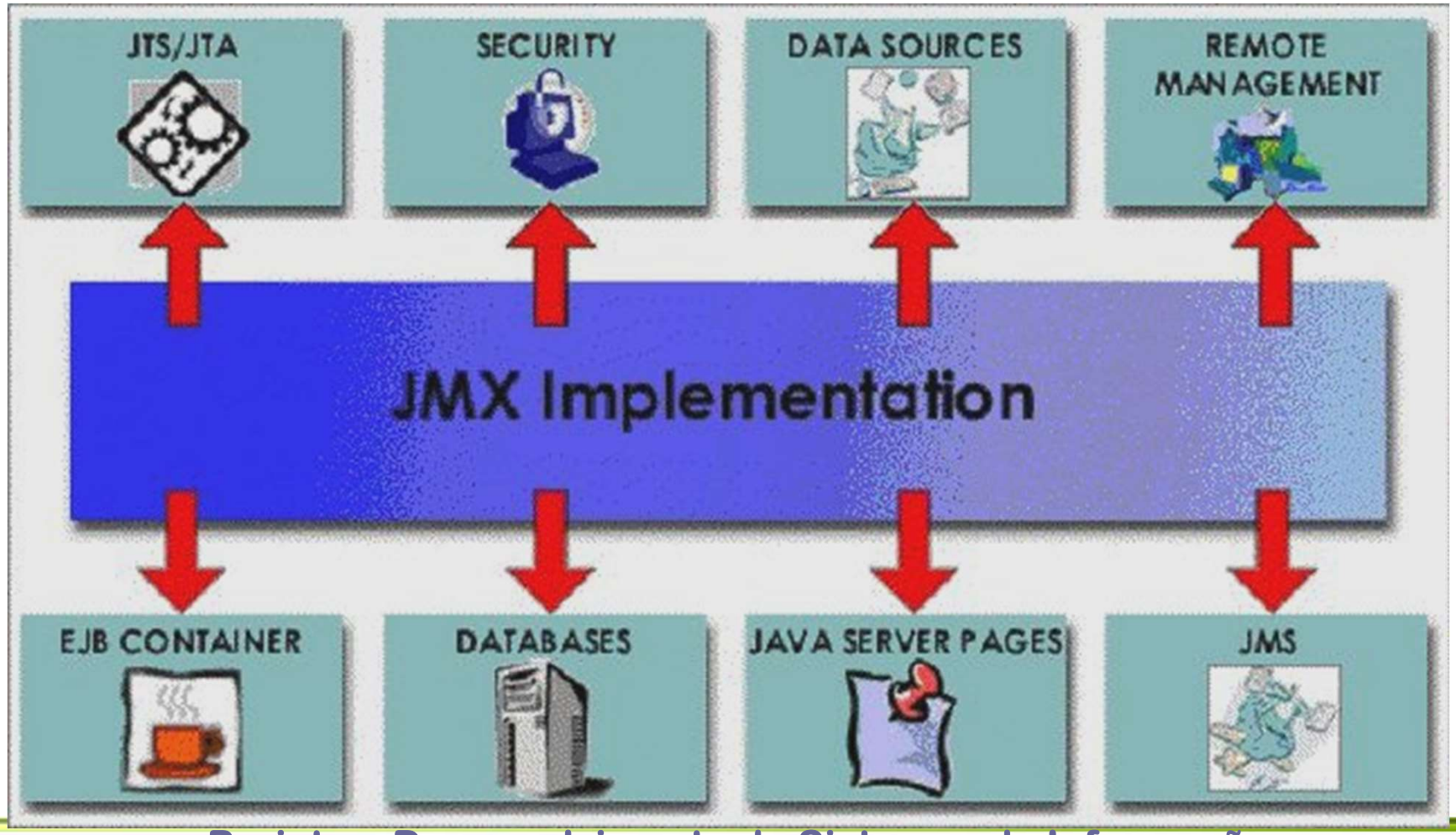
Servidor externo requisitando serviço



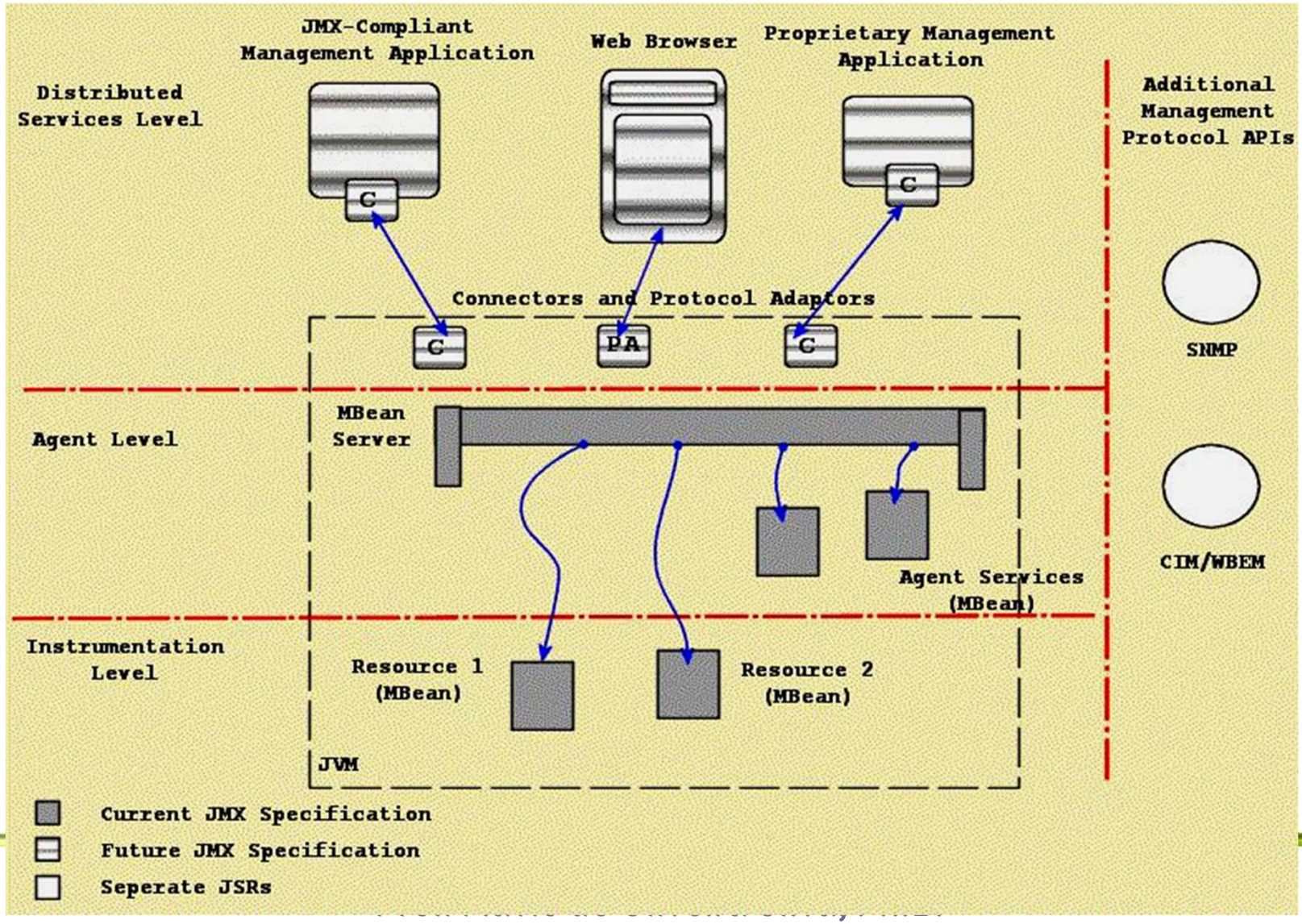
Monolítico x Microkernel



JBoss - JMX

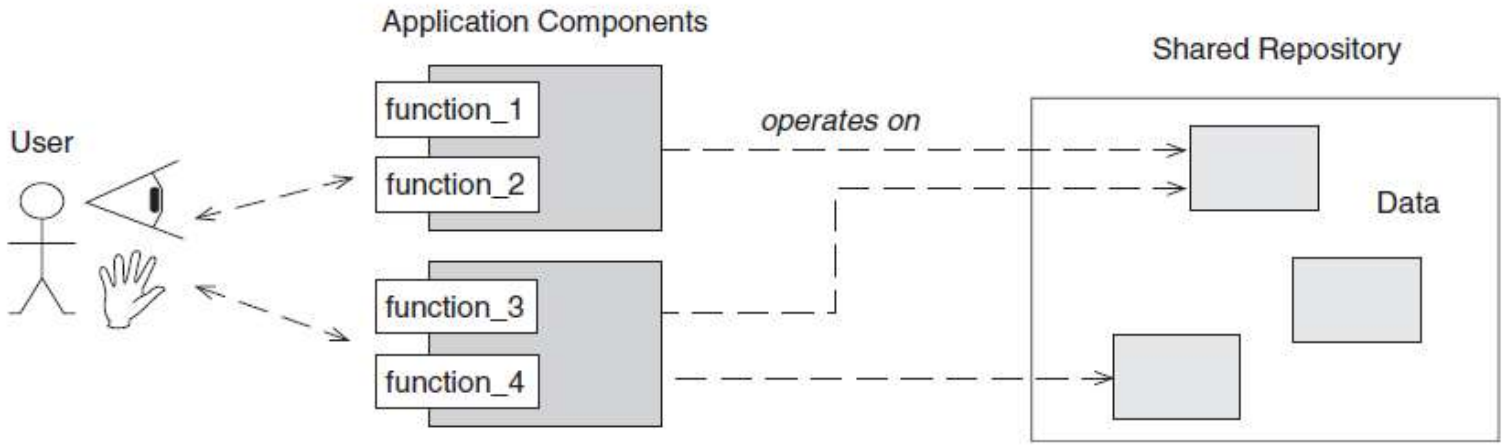

JTS/JTA
SECURITY
DATA SOURCES
REMOTE MANAGEMENT
JMX Implementation
EJB CONTAINER
DATABASES
JAVA SERVER PAGES
JMS

JBoss – JMX
Camadas - Microkernel



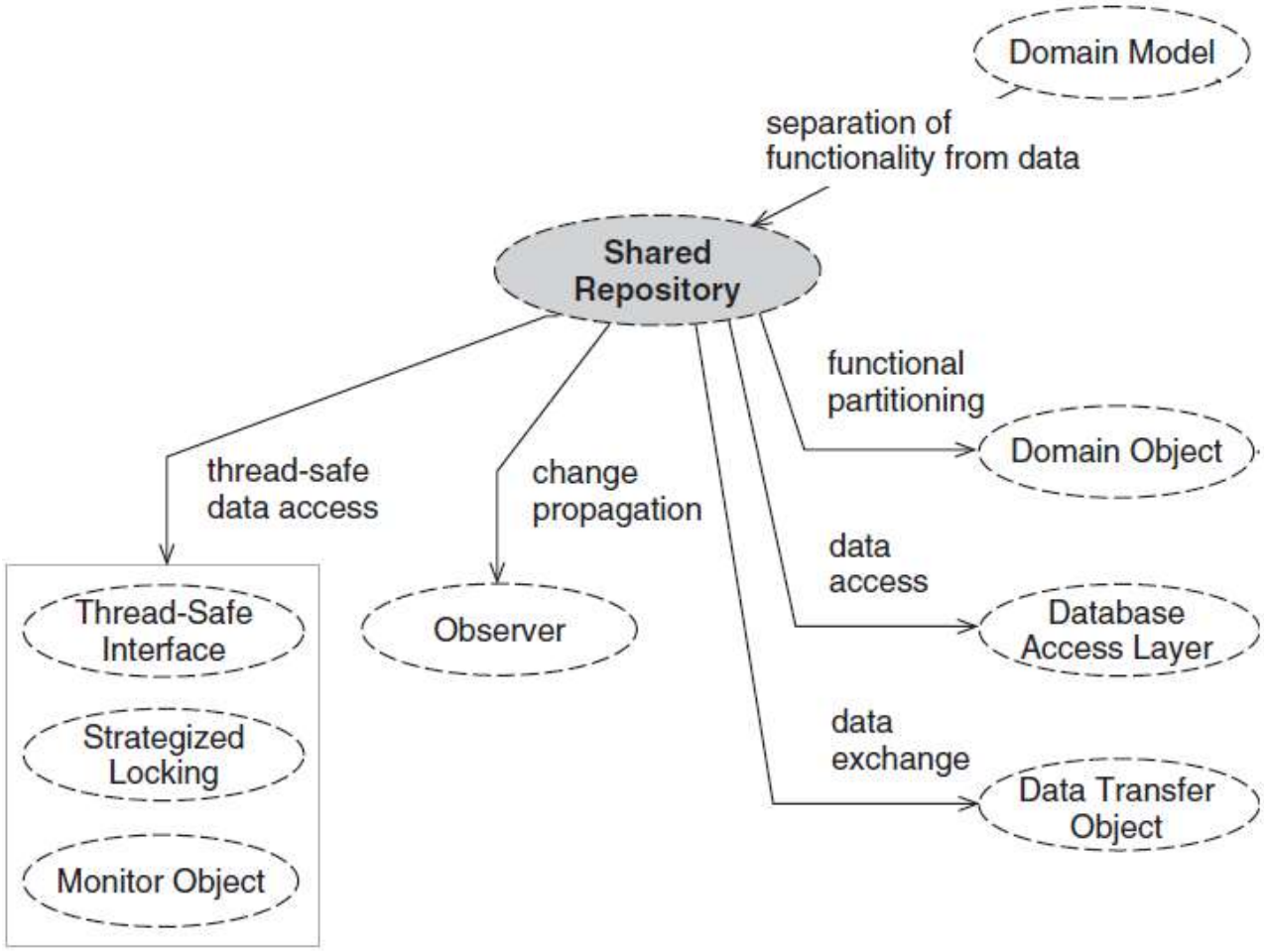
Shared Repository

- Uso
  - Aplicações em que partes operam e são coordenadas através de um conjunto de dados compartilhados
  - Caso de aplicações orientadas a dados e onde a interação entre os componentes não seguem regras específicas que possibilitem sua conexão
  - A conexão entre os componentes e aplicações é feita através dos dados



Shared Repository

- Uso com outros padrões



Documento de Arquitetura x Entregas

- Seção 2. Representação Arquitetural
  - Integrações e Protótipo da Arquitetura (D3)
- Seção 4. Visão de Casos de Uso
  - Protótipo de telas (D1)
- Seção 5. Visão Lógica
  - Diagramas com visão estrutural dos casos de uso (D4)
- Seção 6. Visão de Processos
  - Diagramas com visão comportamental dos casos de uso (D4)
- Seção 7. Visão da Implementação
  - Diagrama com visão de implementação do sistema (D4)
- Seção 8. Visão da Implantação
  - Diagramas com visão de implantação do sistema (D4)
- Seção 9. Visão de Dados
  - Persistência Dados (D2)

Persistência de Dados

- Dados de curta duração
  - Bancos de Dados NoSQL
  - Orion Publish/Subscribe Context Broker
- Dados de longa duração
  - Banco de Dados Relacional
- Dados Históricos

Protótipo Arquitetura

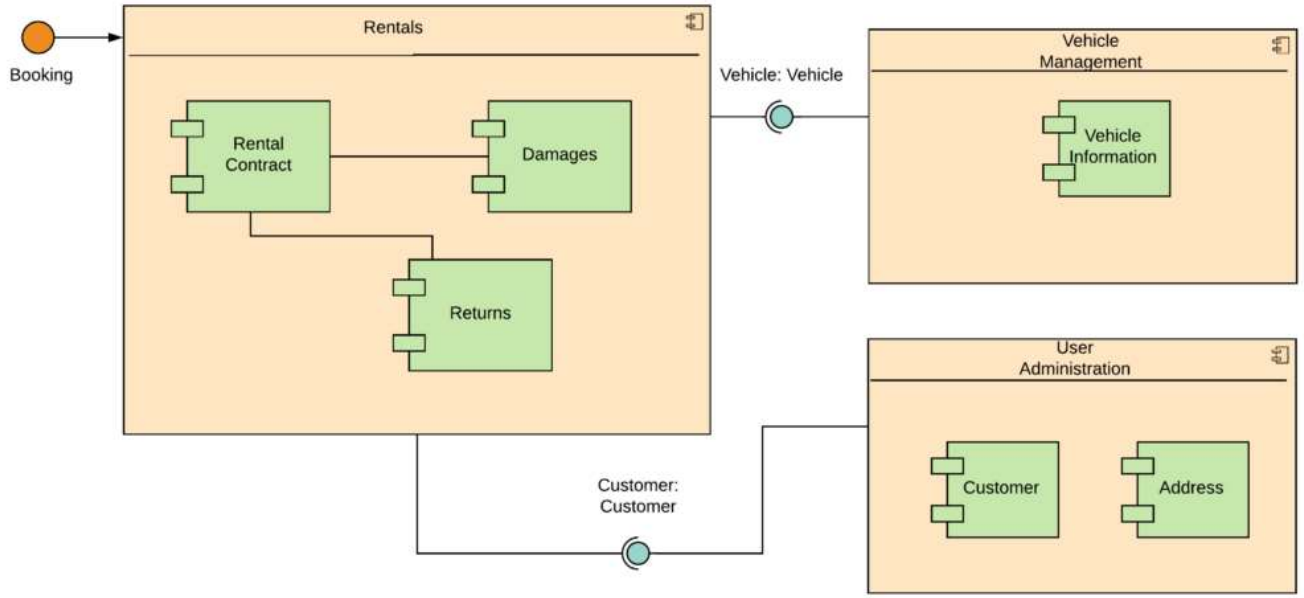
- Diagramas de mais alto nível
- Mostram os principais componentes do sistema (foco na estrutura) e suas comunicações (foco no comportamento)
- Diagramas
  - Componentes, Pacote ou Classes Gerais (estrutura)
  - Sequencia ou Comunicação (Comportamento)
- Mostram a solução genérica que será utilizada pelo Sistema
  - Neste caso a arquitetura pode ser mostrad através de classes genéricas
  - Os casos de uso serão criados como casos particulares deste visão genérica
  - Exemplo: CView (uma classe genérica com a visualização) e CUserView e CProductView duas classes que são especificas dos casos de uso ligados a usuário e produtos

Integrações

- Integrações
  - Externas
    - Componentes ou Serviços estão fora da gestão do proprietário do software
    - Google Maps; Facebook Login
  - Internas
    - Componentes ou Serviços dentro fora da gestão do proprietário do software mas não foram criados pelo mesmo proprietário
    - Componente Java-JDBC; DBMS
  - Comunicações (http; api calls)

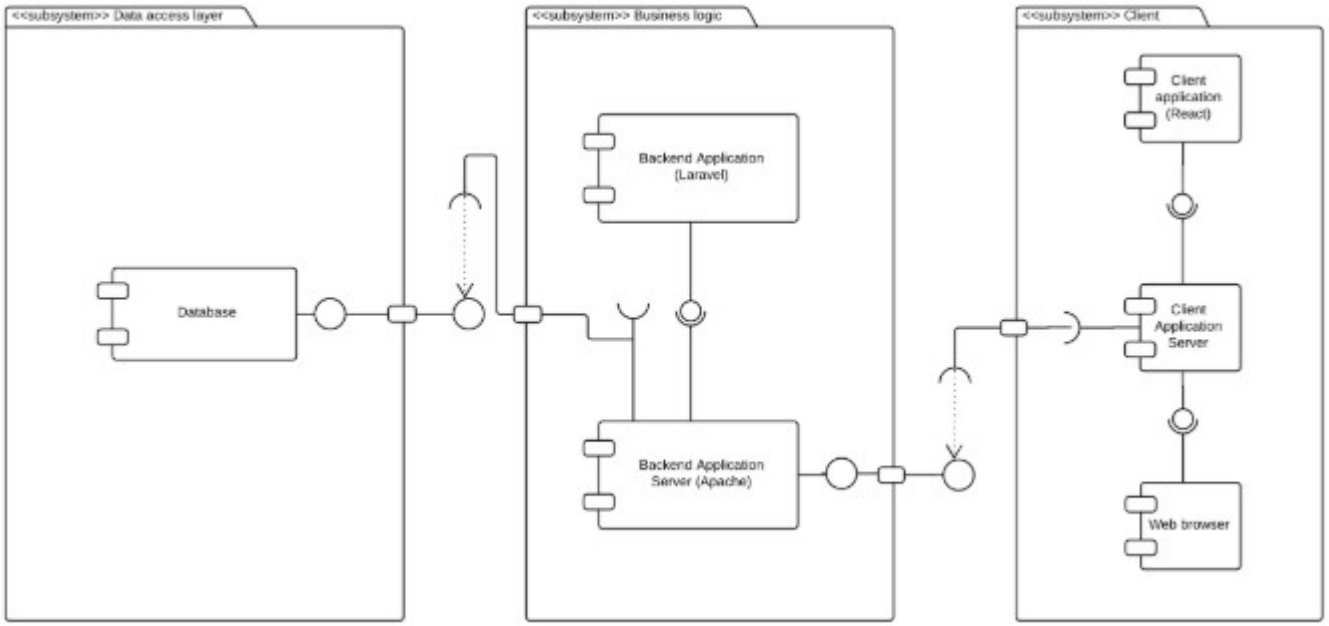
Prótótipo Arquiteutra Estrutura - Exemplo

- Diagrama de Componentes



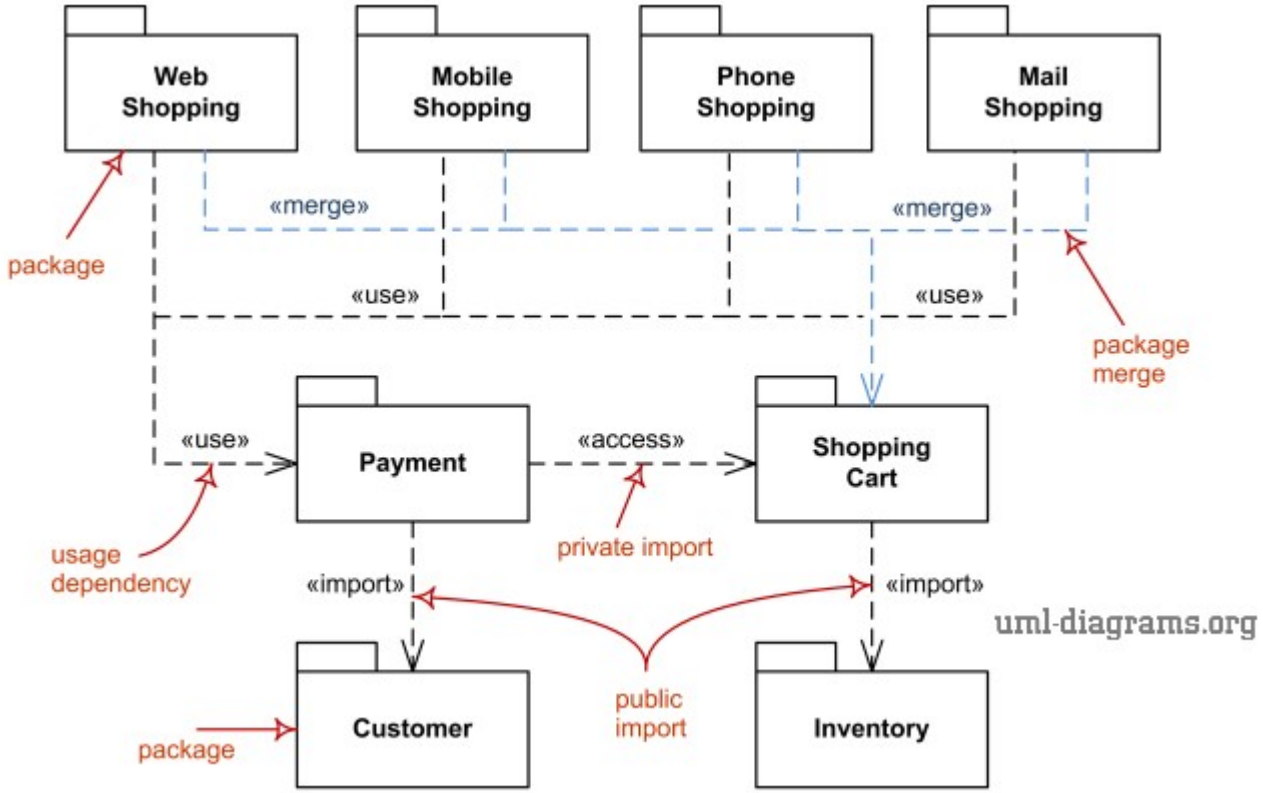
Prótótipo Arquiteutra Estrutura - Exemplo

- Diagrama de Componentes



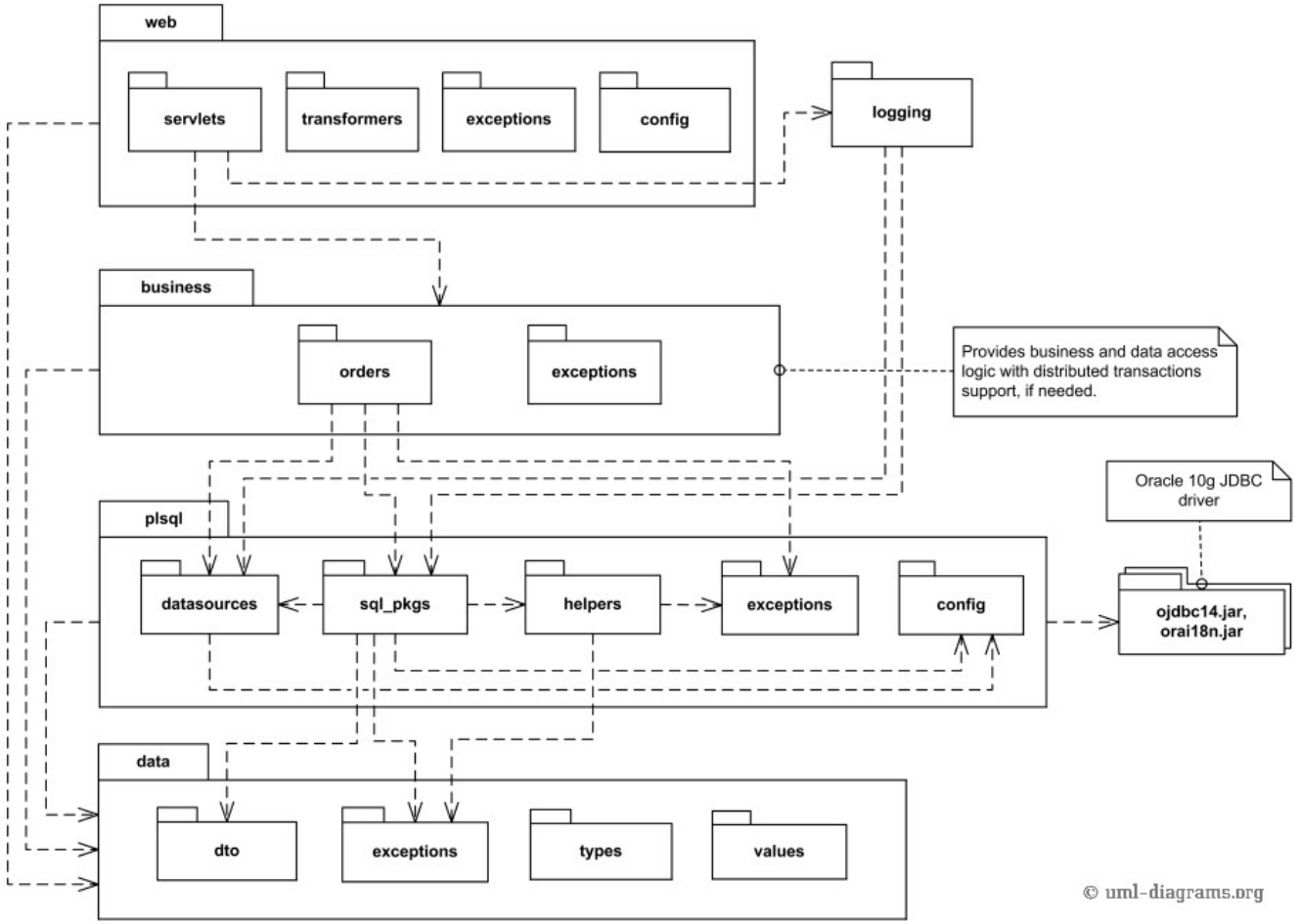
Prótótipo Arquiteutra Estrutura - Exemplo

- Diagrama de Pacotes



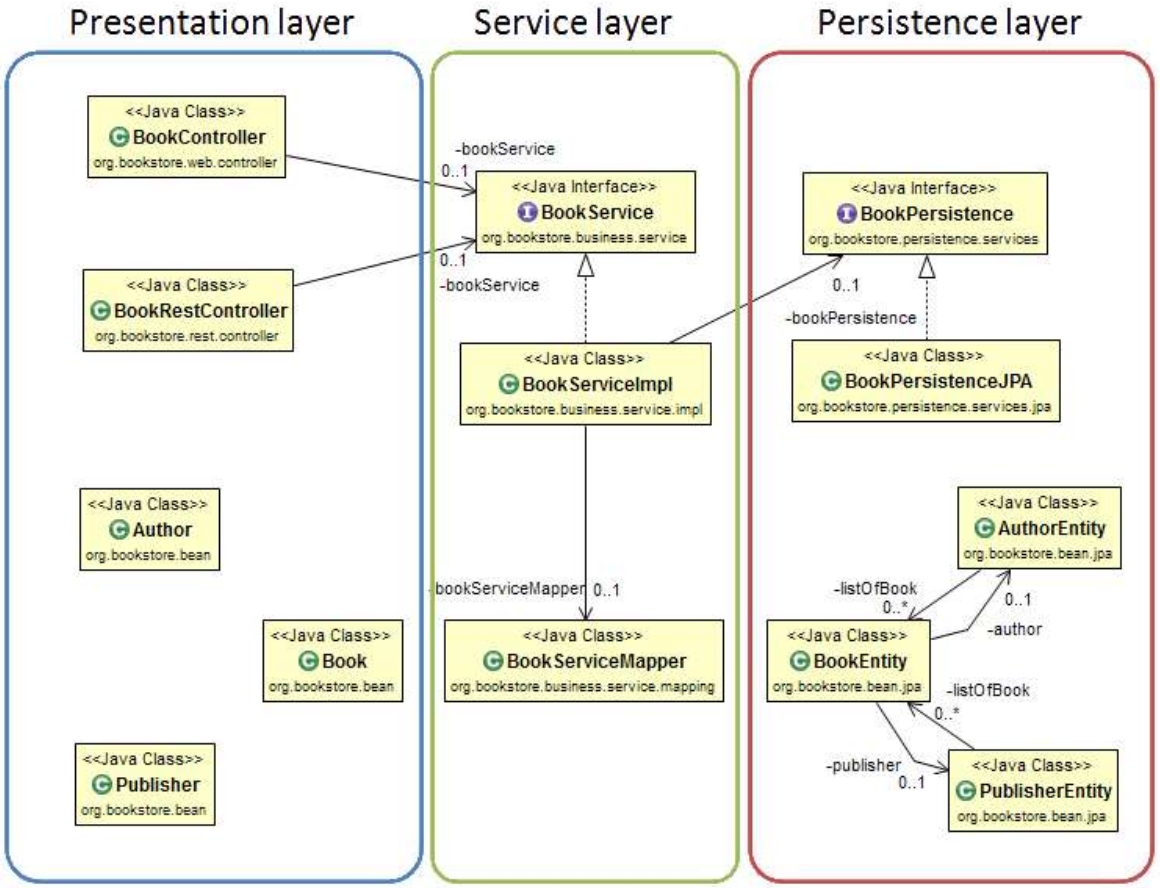
Prótótipo Arquiteutra Estrutura - Exemplo

- Diagrama de Pacotes



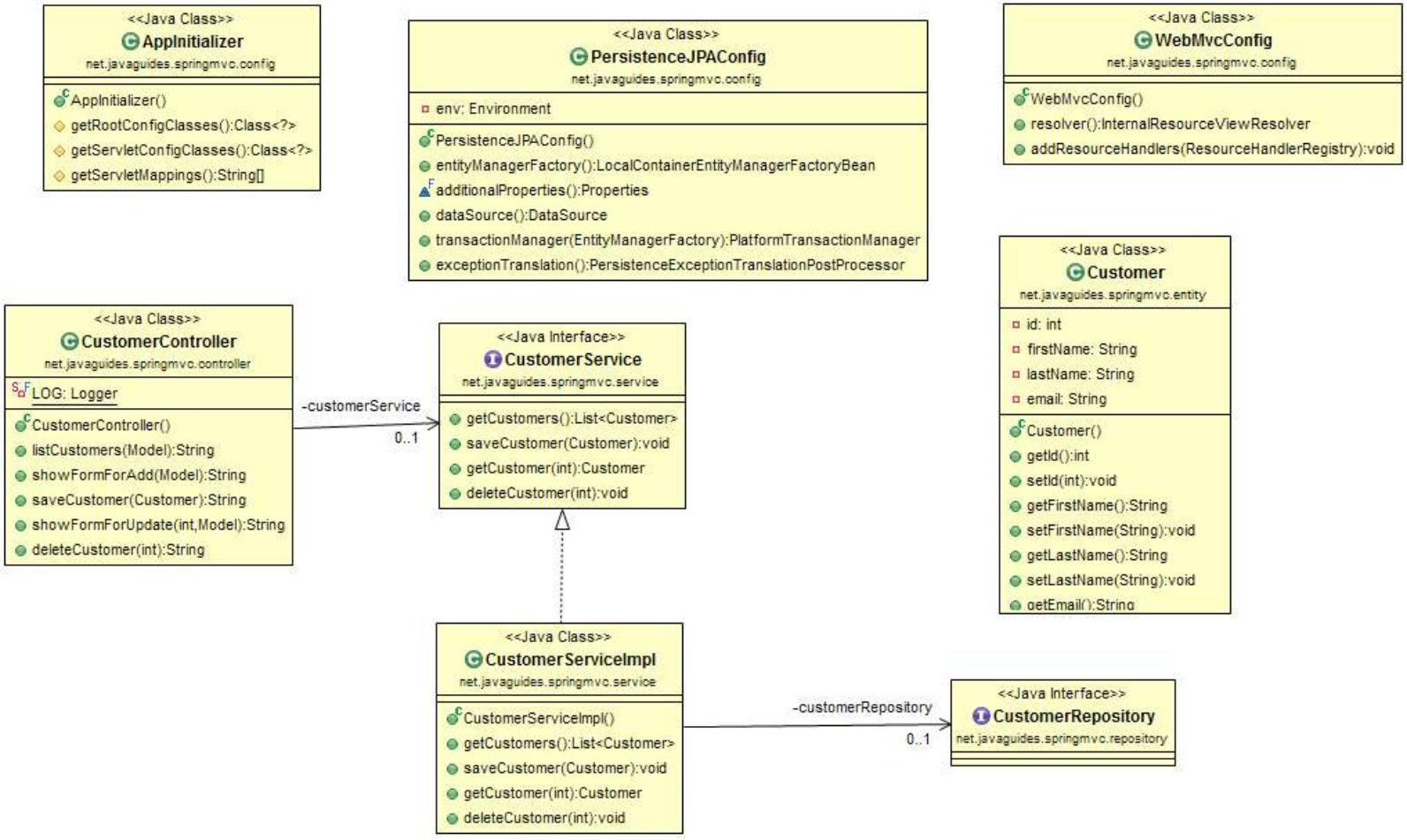
Prótótipo Arquiteutra Estrutura - Exemplo

- Diagrama de Classes



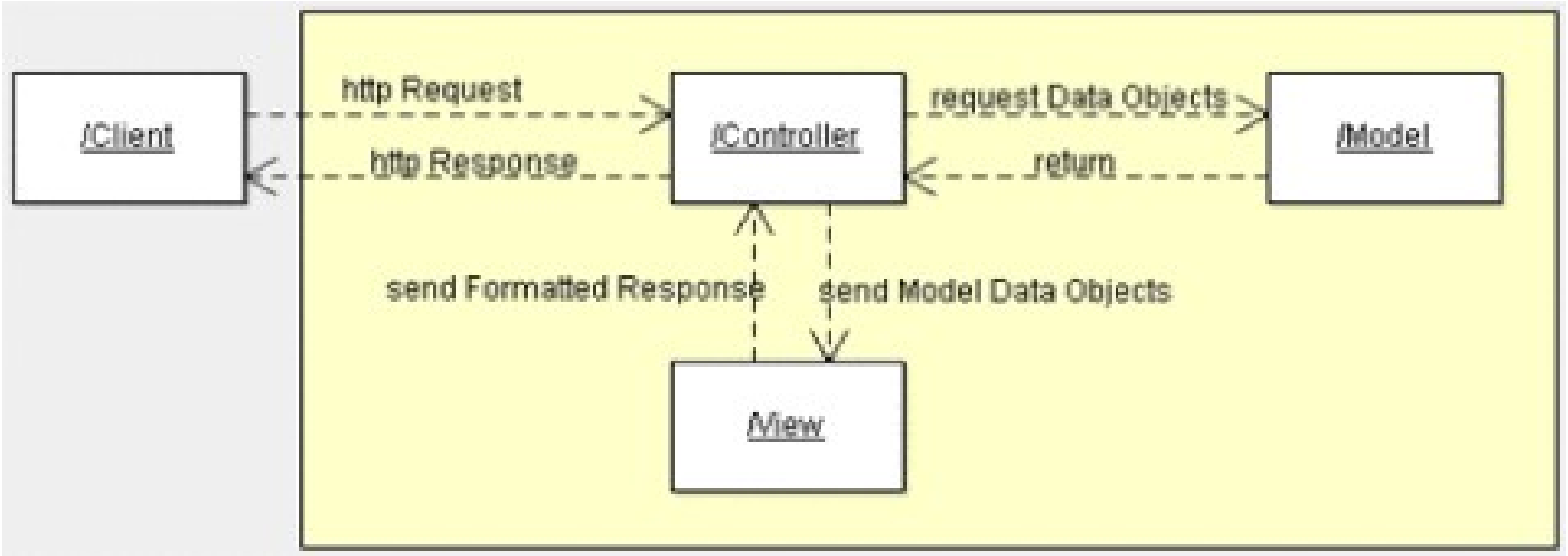
Prótótipo Arquiteutra Estrutura - Exemplo

- Diagrama de Classes



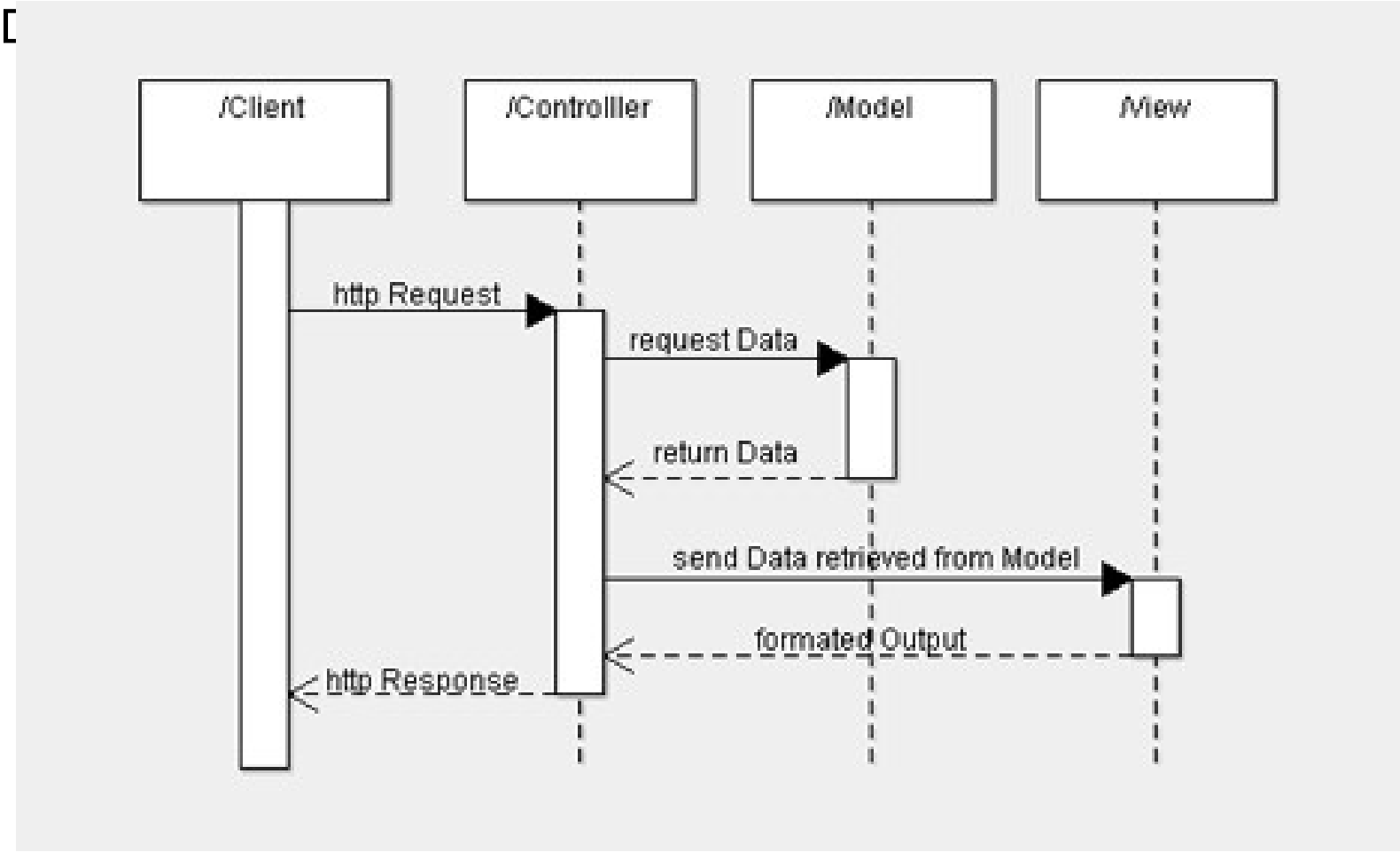
Prótótipo Arquitetutra Comprtamento - Exemplo

- Diagrama de Classes genérico



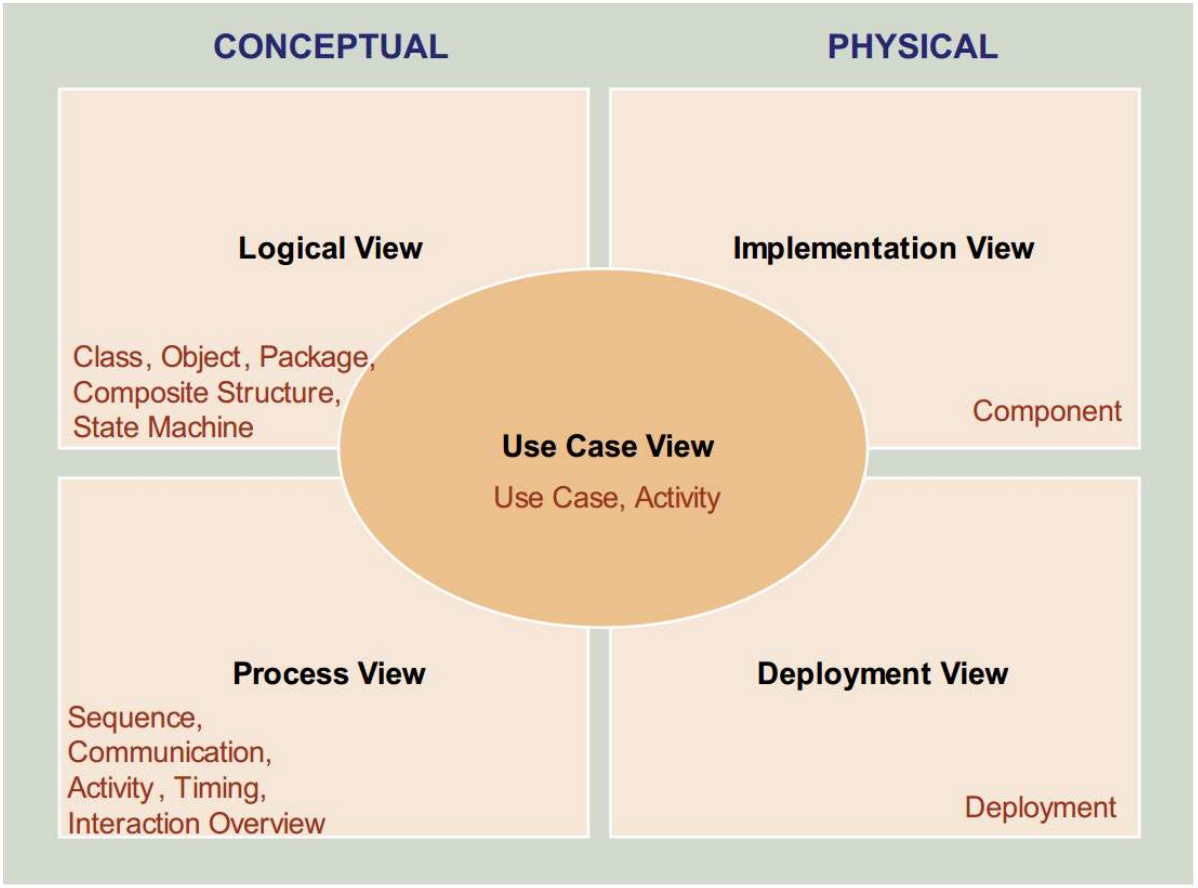
Prótótipo Arquitetutra Estrutura - Exemplo

- [



Visões do Sistema
Diagramas UML

- Diferentes visões do mesmo software (Sistema)

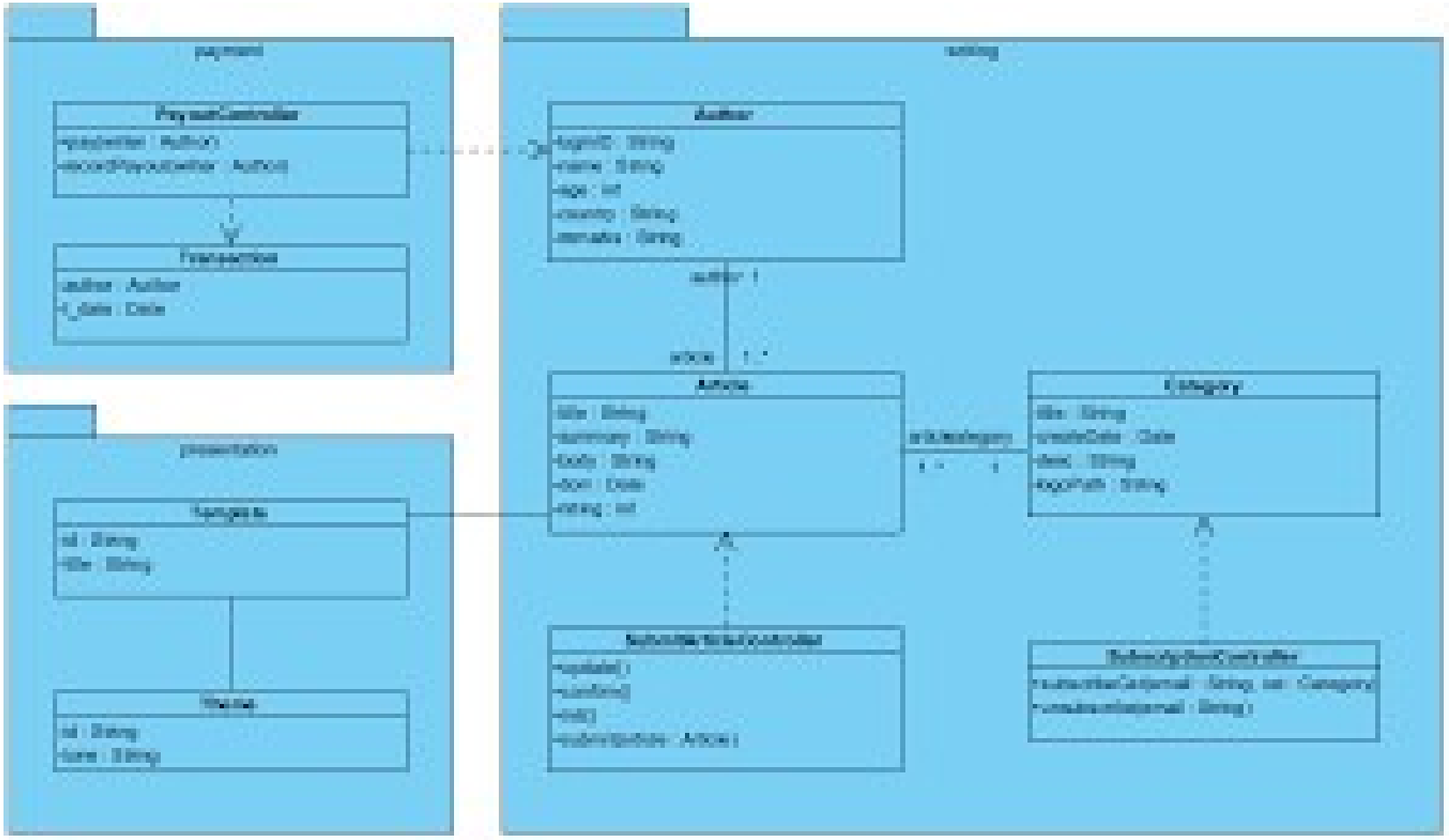


http://www.sparxsystems.com/downloads/whitepapers/FCGSS_US_WP_Applying_4+1_w_UML2.pdf

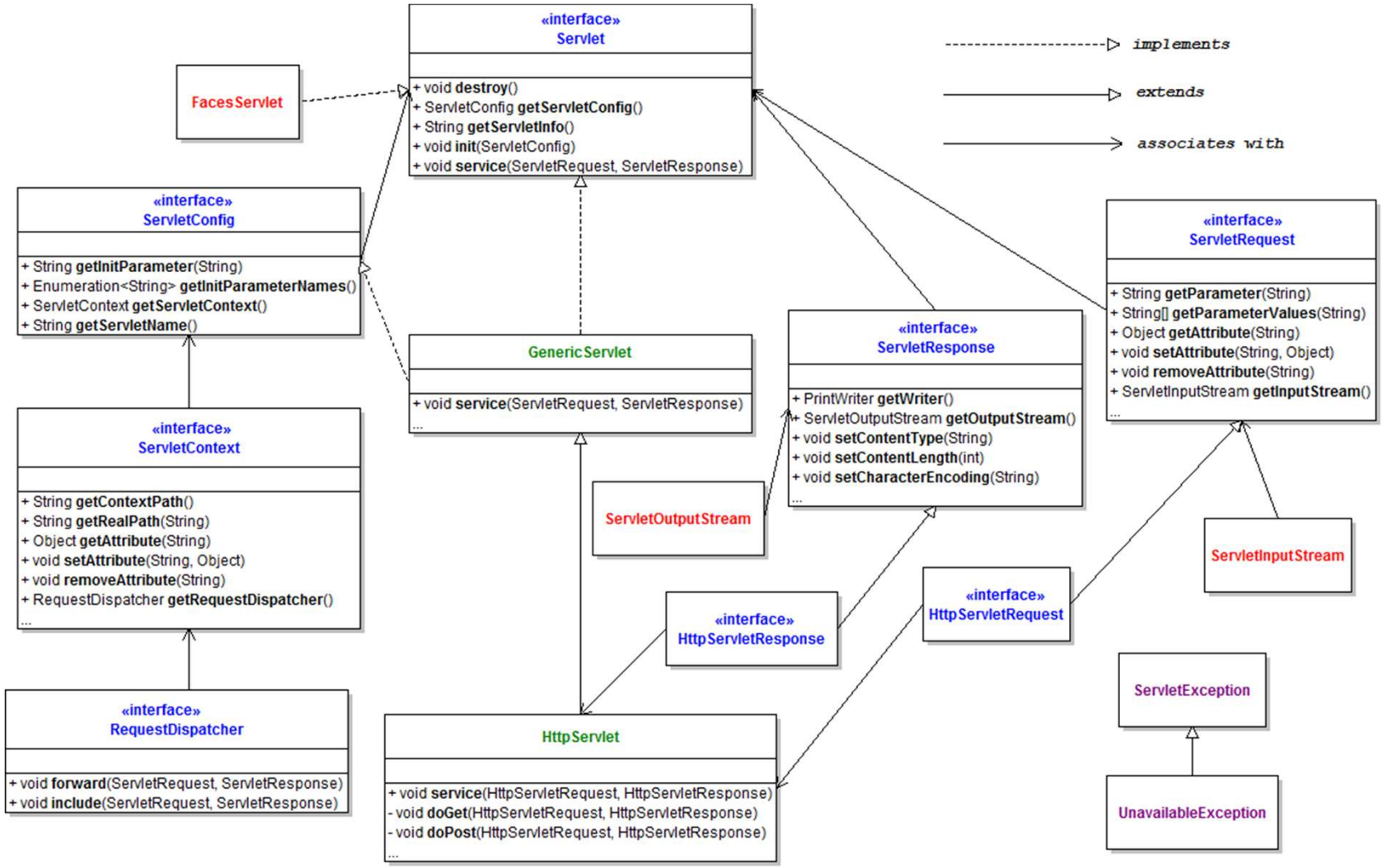
Visões do Sistema Diagramas UML

- Cenários (Use Case View ou Scenarios view)
  - Apresenta uma visão próxima do usuário, descrevendo cenários de uso da aplicação
  - Normalmente é a primeira visão construída
  - Diagramas: Casos de Uso
- Lógica (Logical View)
  - Foco nas funcionalidades, normalmente dividida em subsistemas apresentando a estrutura (classes) envolvidas
  - Diagramas: Componentes, Pacotes, Classe (no geral diagramas estruturais)
- Processos (Process View)
  - Foco em aspectos não funcionais
  - Diagramas: Atividade, Sequencia, Comunicação (no geral diagramas comportamentais)
- Implementação (Implementation View)
  - Útil para gestão de configuração
  - Apresenta pacotes e componentes que serão implantados
  - Diagramas: Componente ou Pacotes
- Implantação (Deployment View)
  - Consiste do mapeamento do software no hardware onde será implantado
  - Diagrama: Implantação

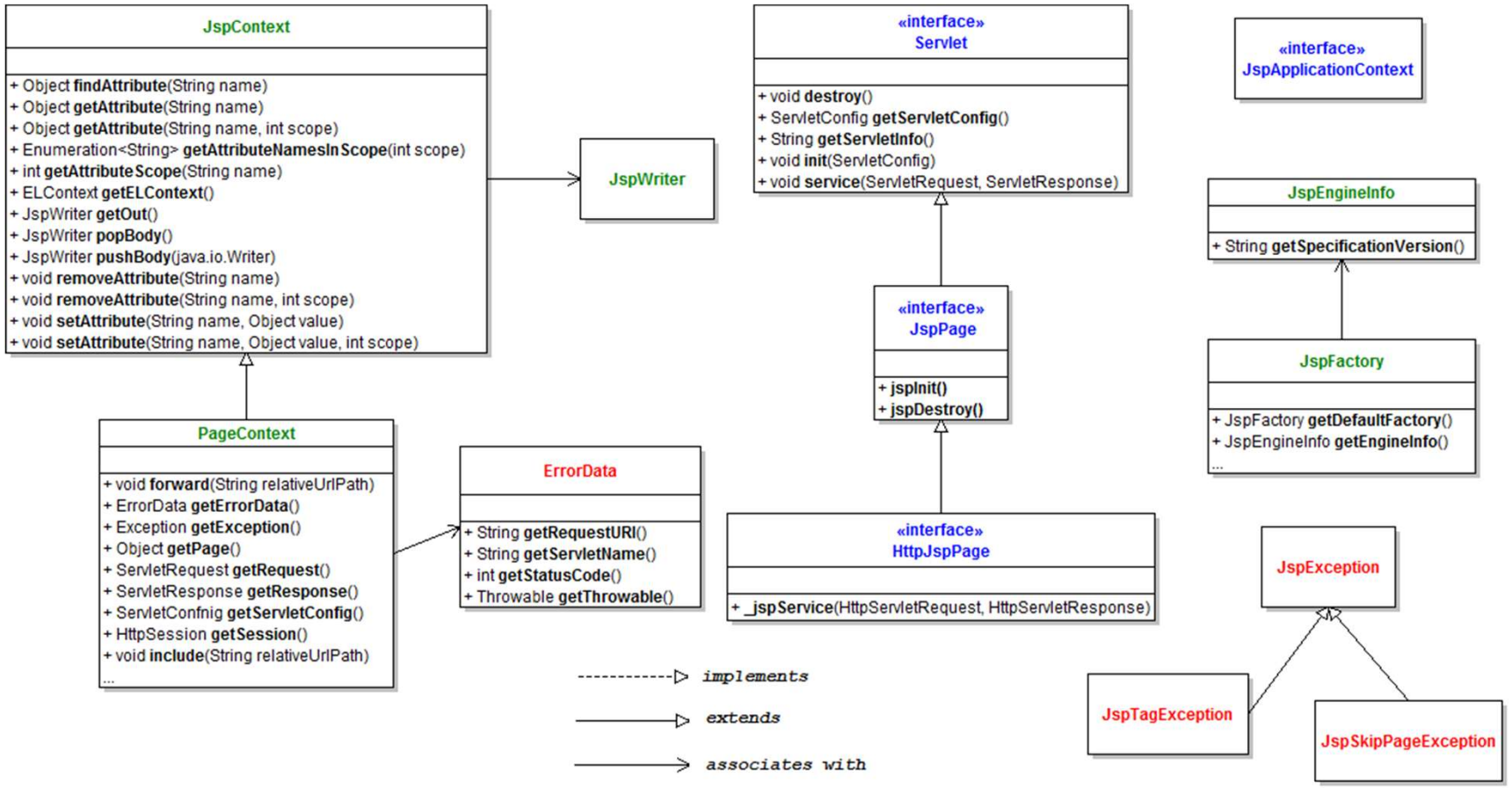
Visão Lógica - Exemplos



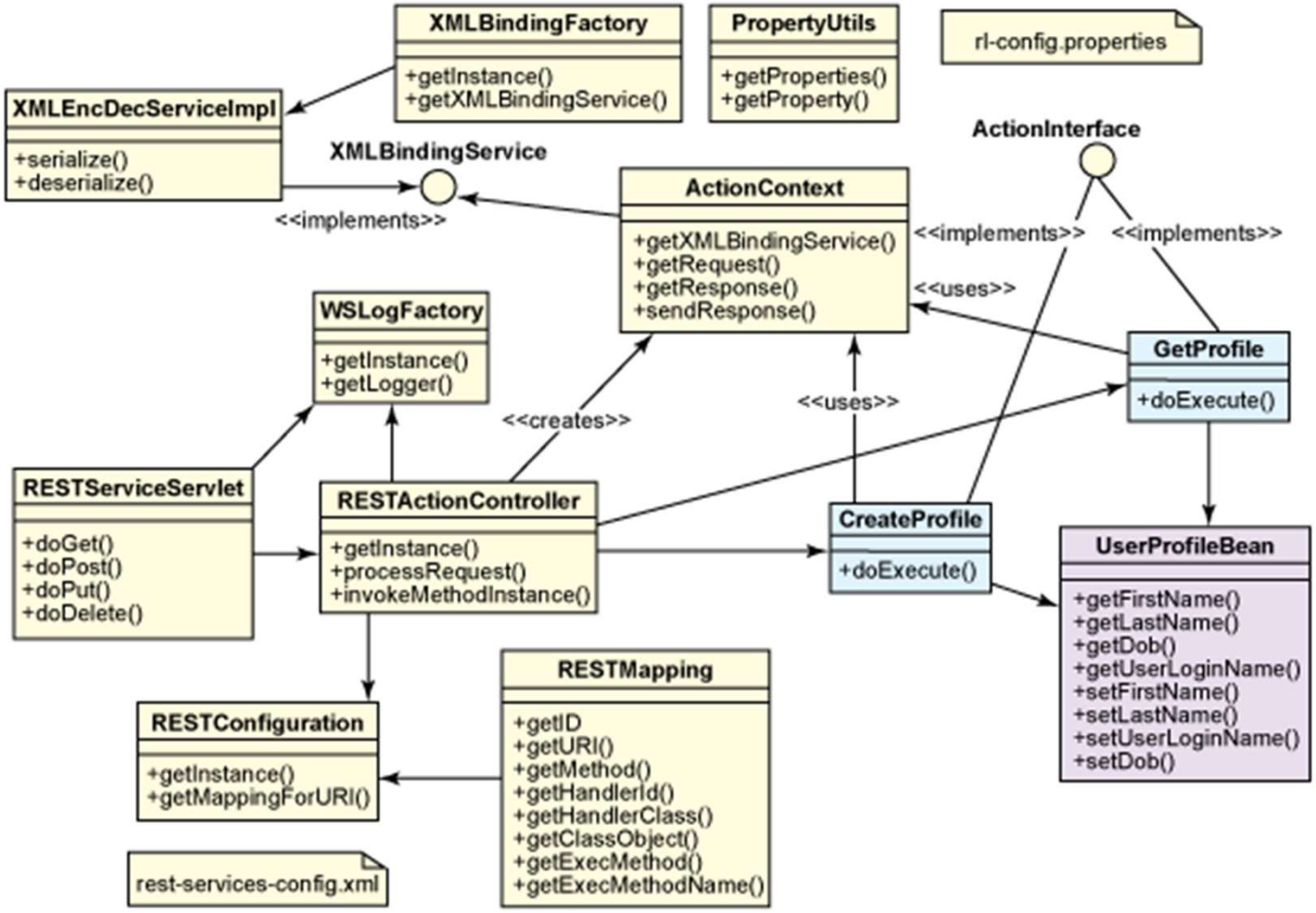
Visão Lógica - Exemplos



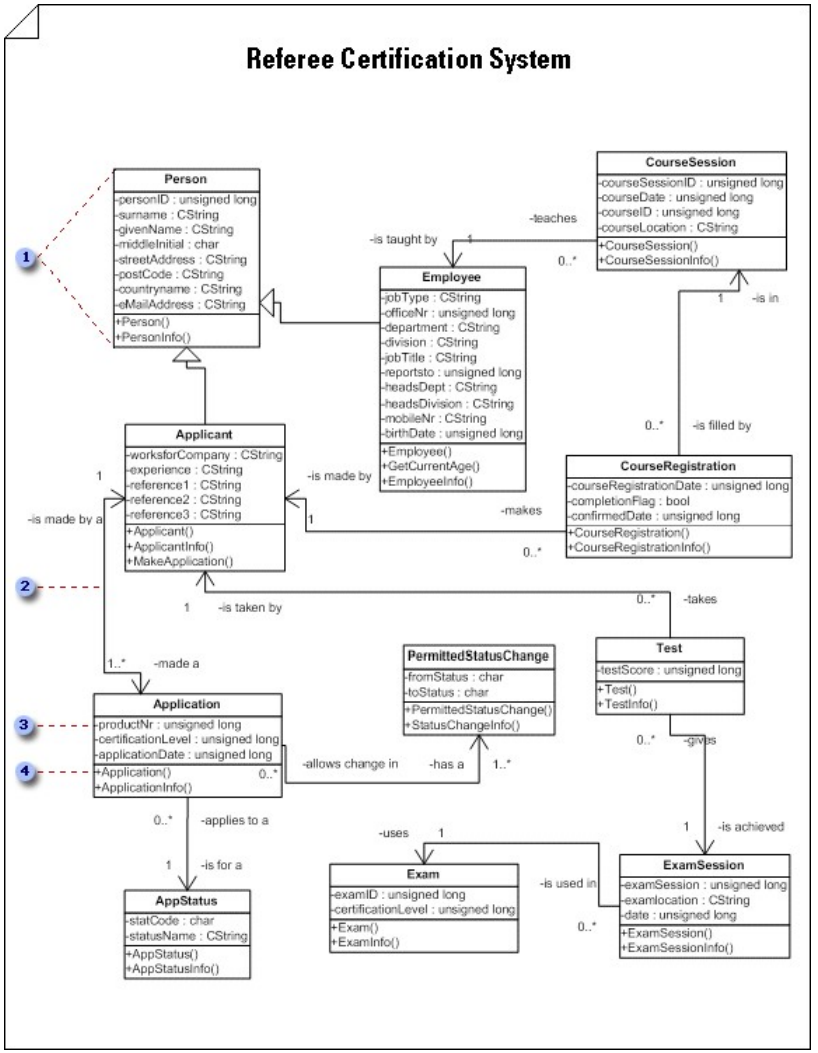
Visão Lógica - Exemplos



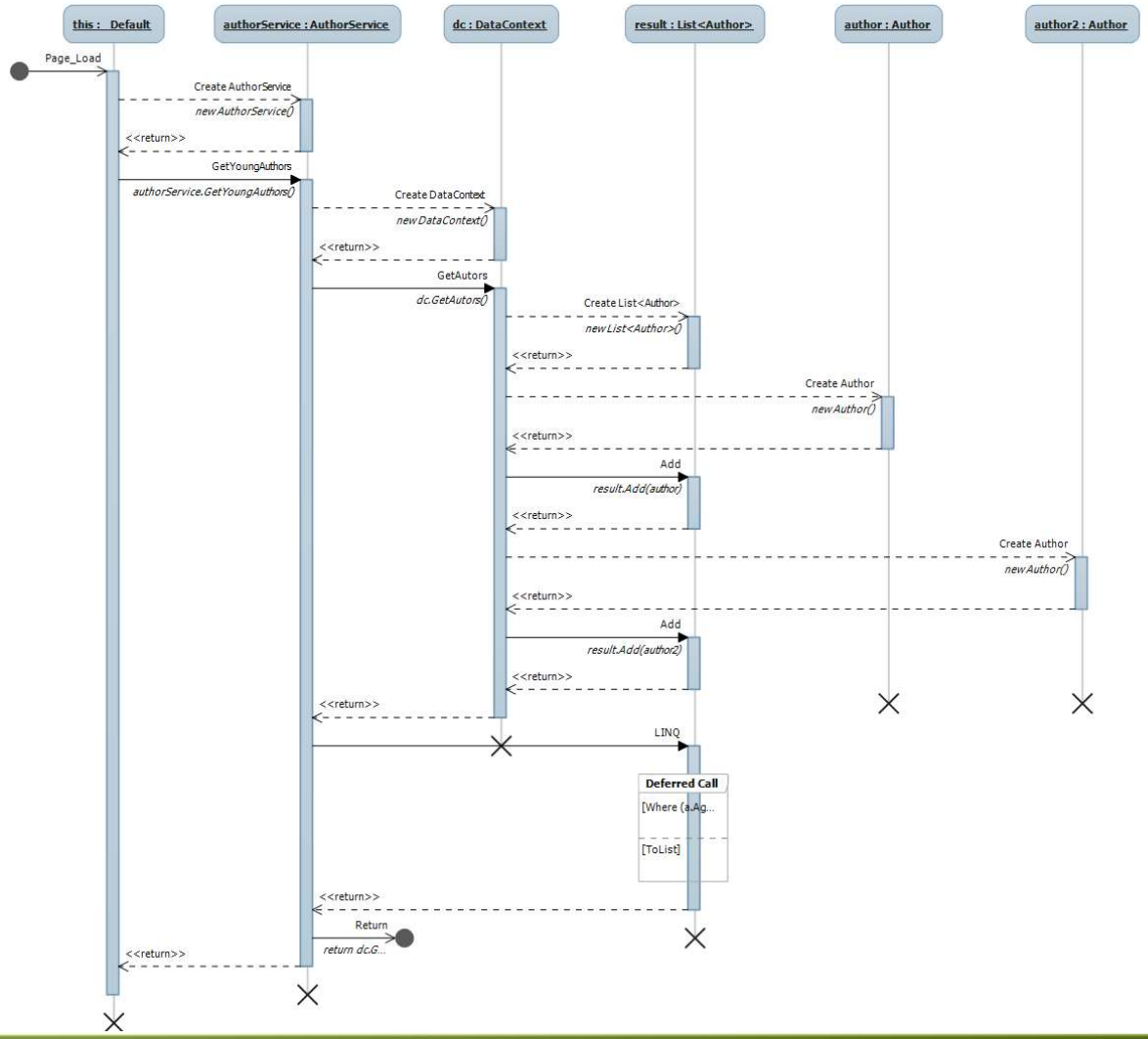
Visão Lógica - Exemplos



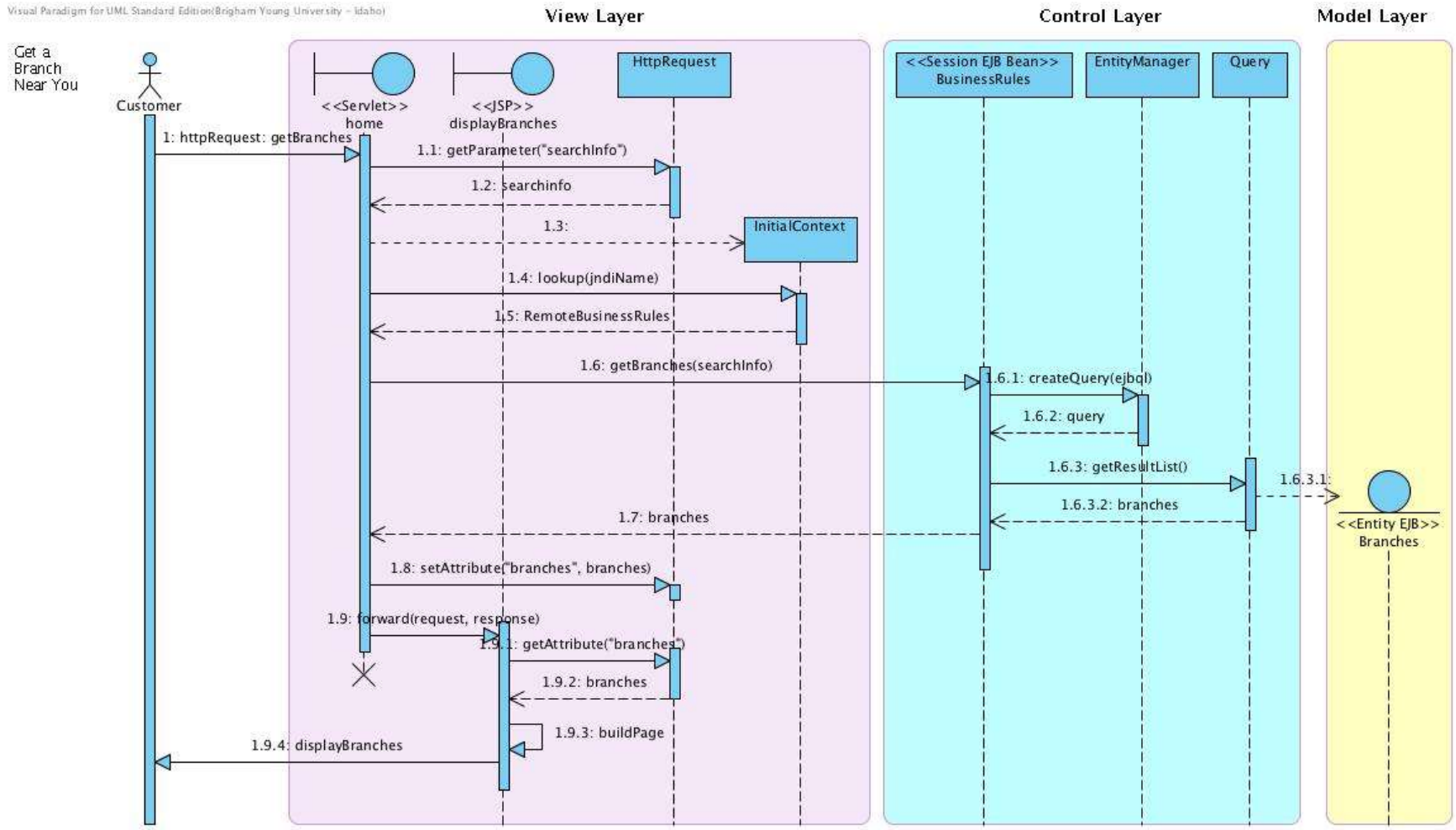
Visão Lógica - Exemplos



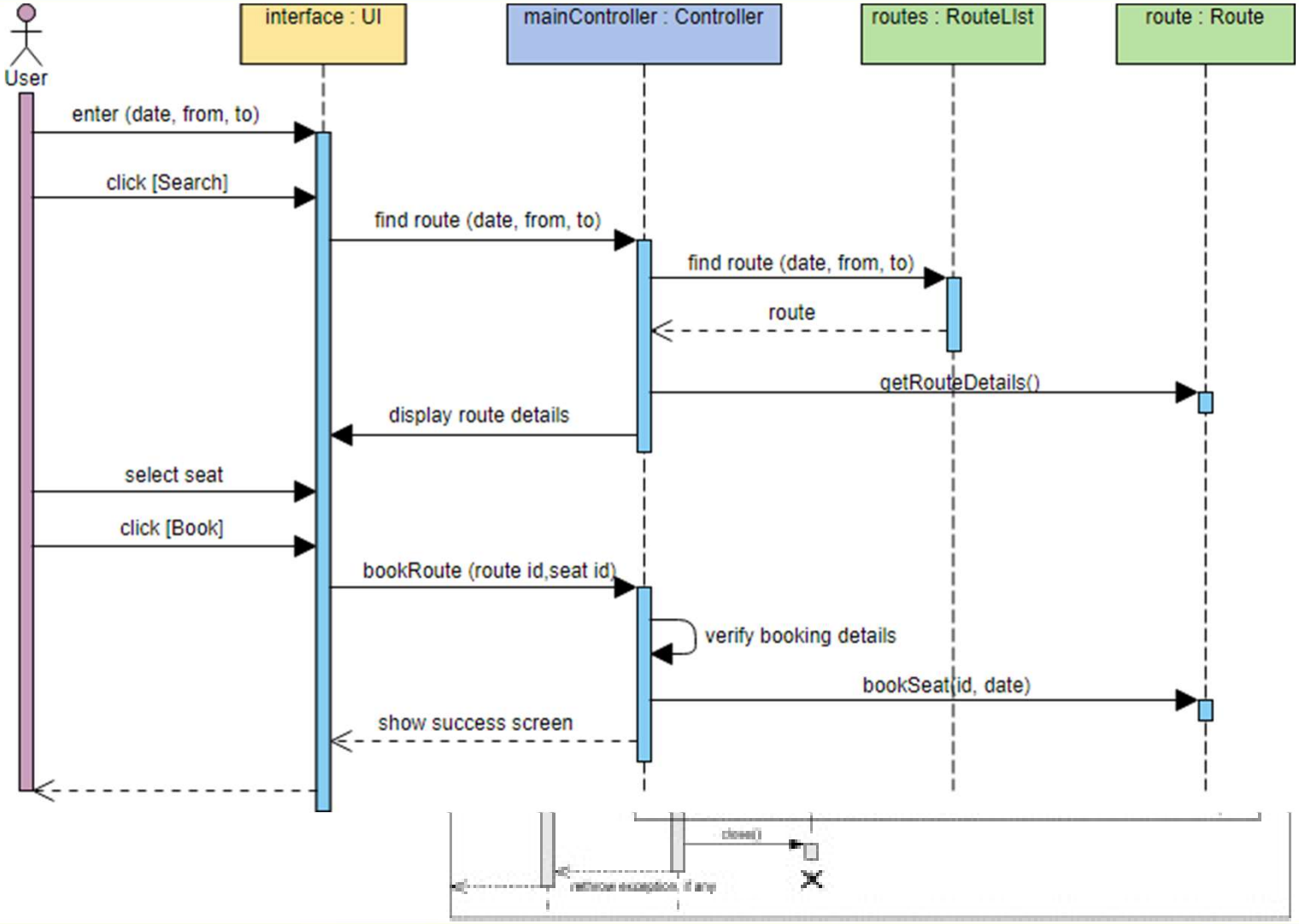
Visão de Processos - Exemplos



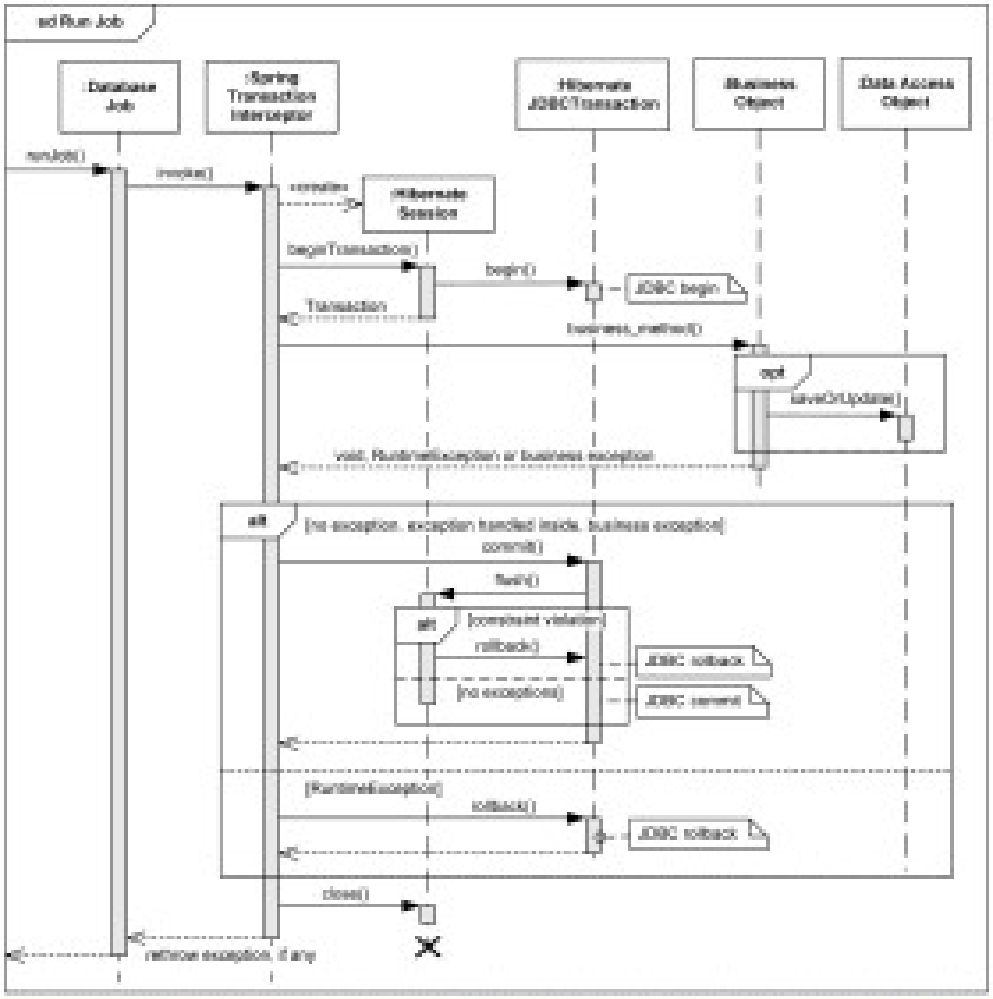
Visão de Processos - Exemplos



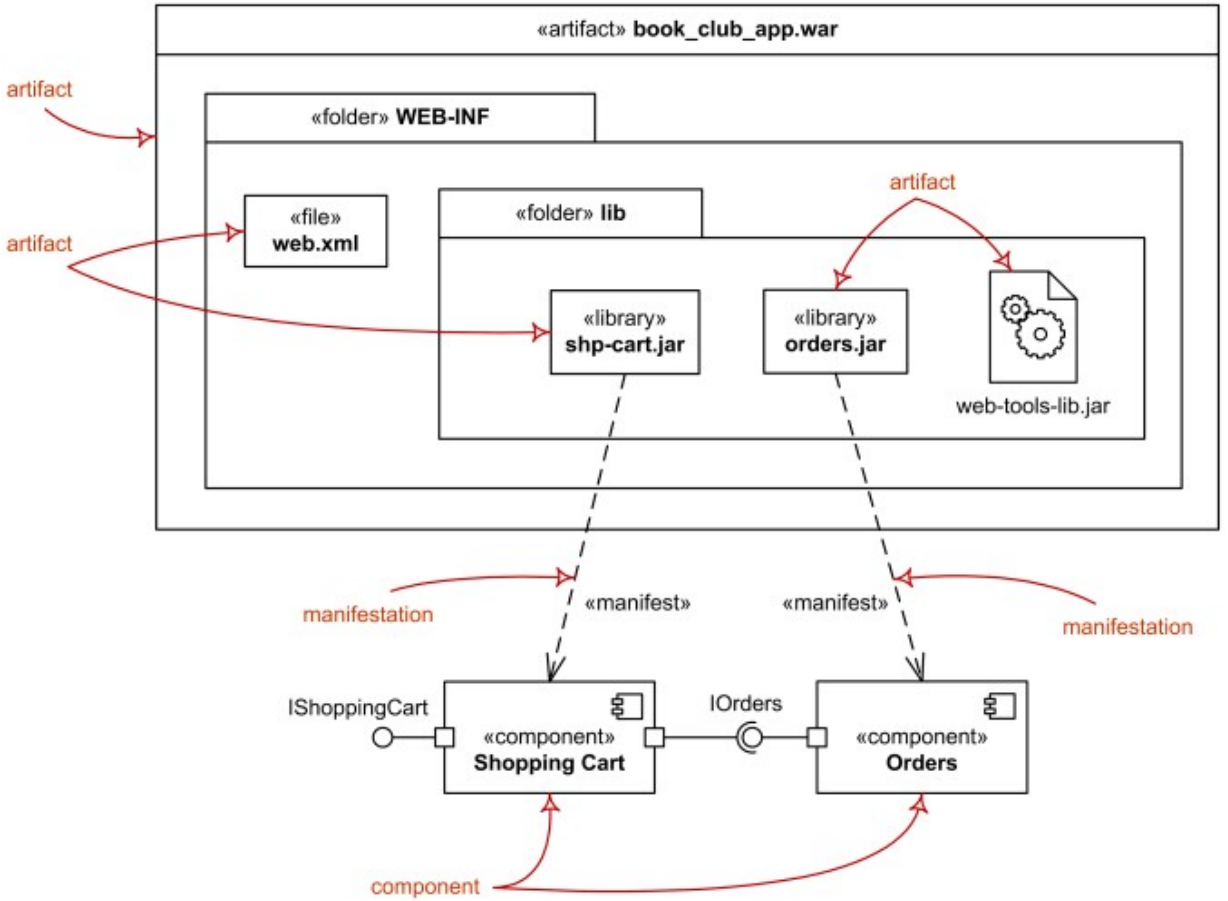
Visão de Processos - Exemplos



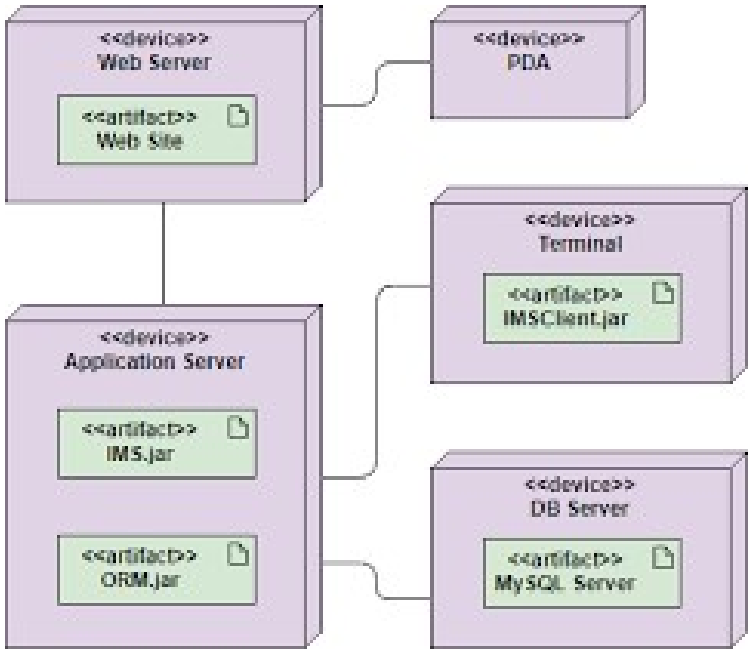
Visão de Processos - Exemplos



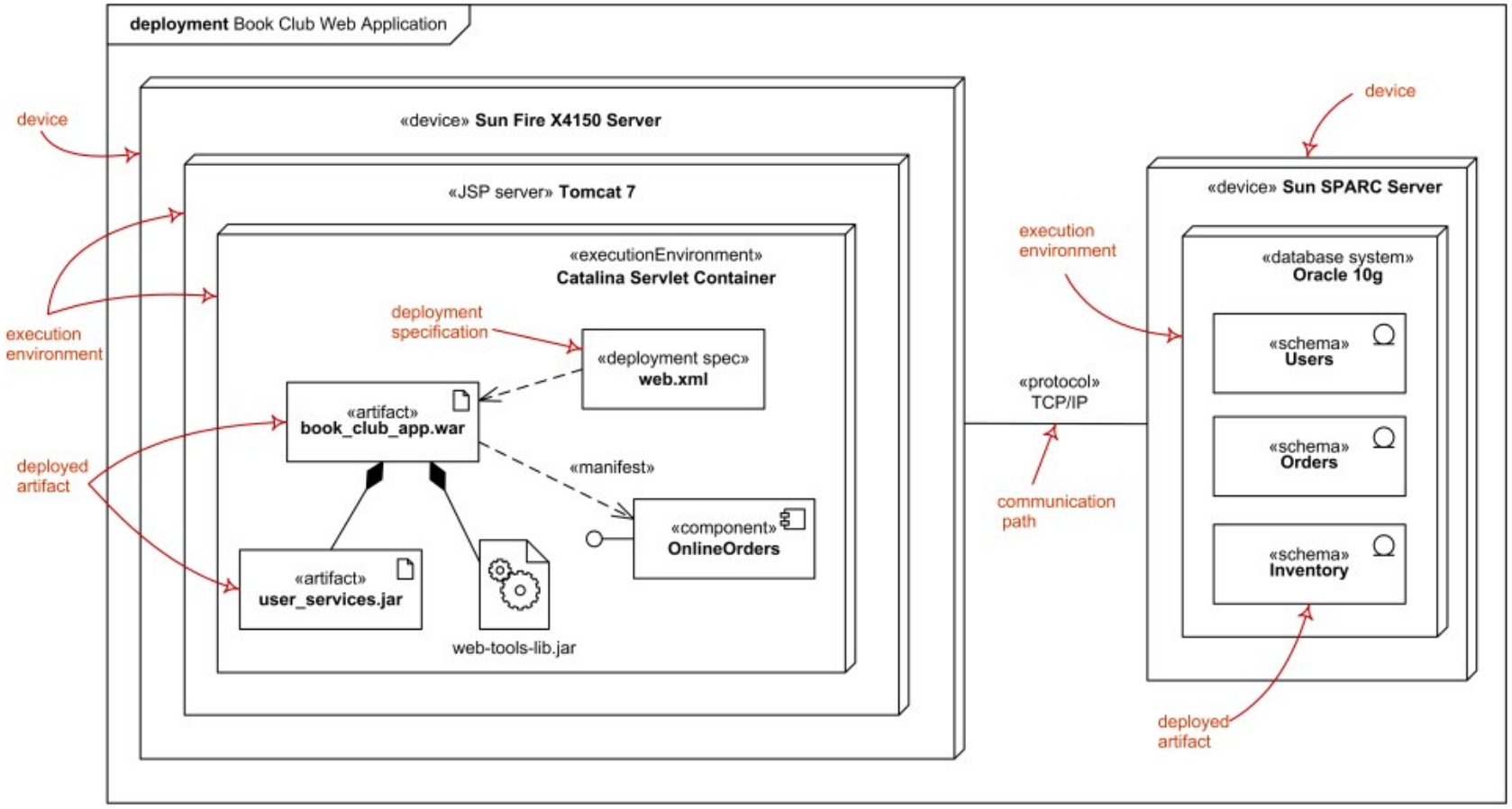
Visão da Implementação – Exemplos



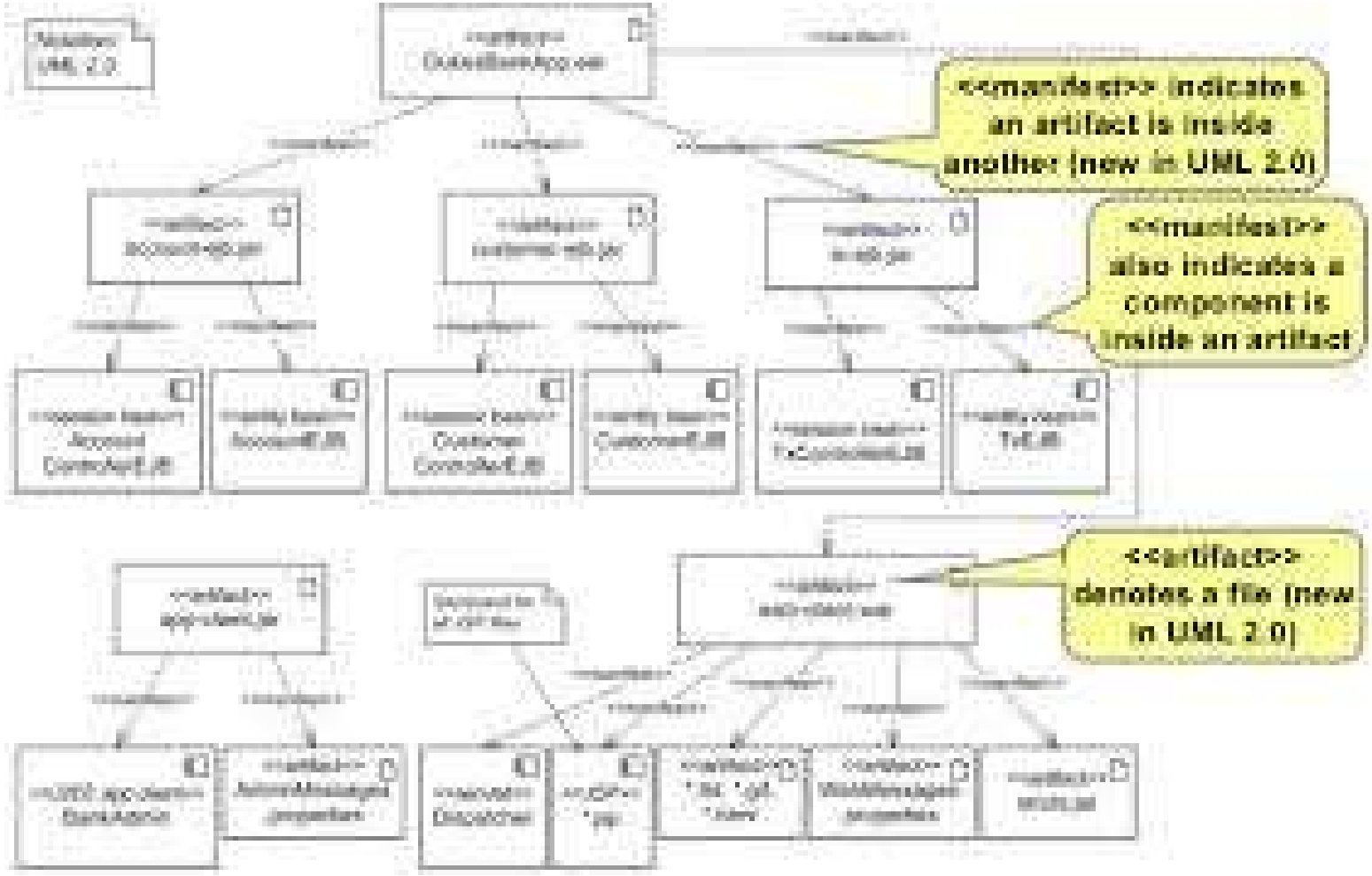
Visão da Implementação – Exemplos



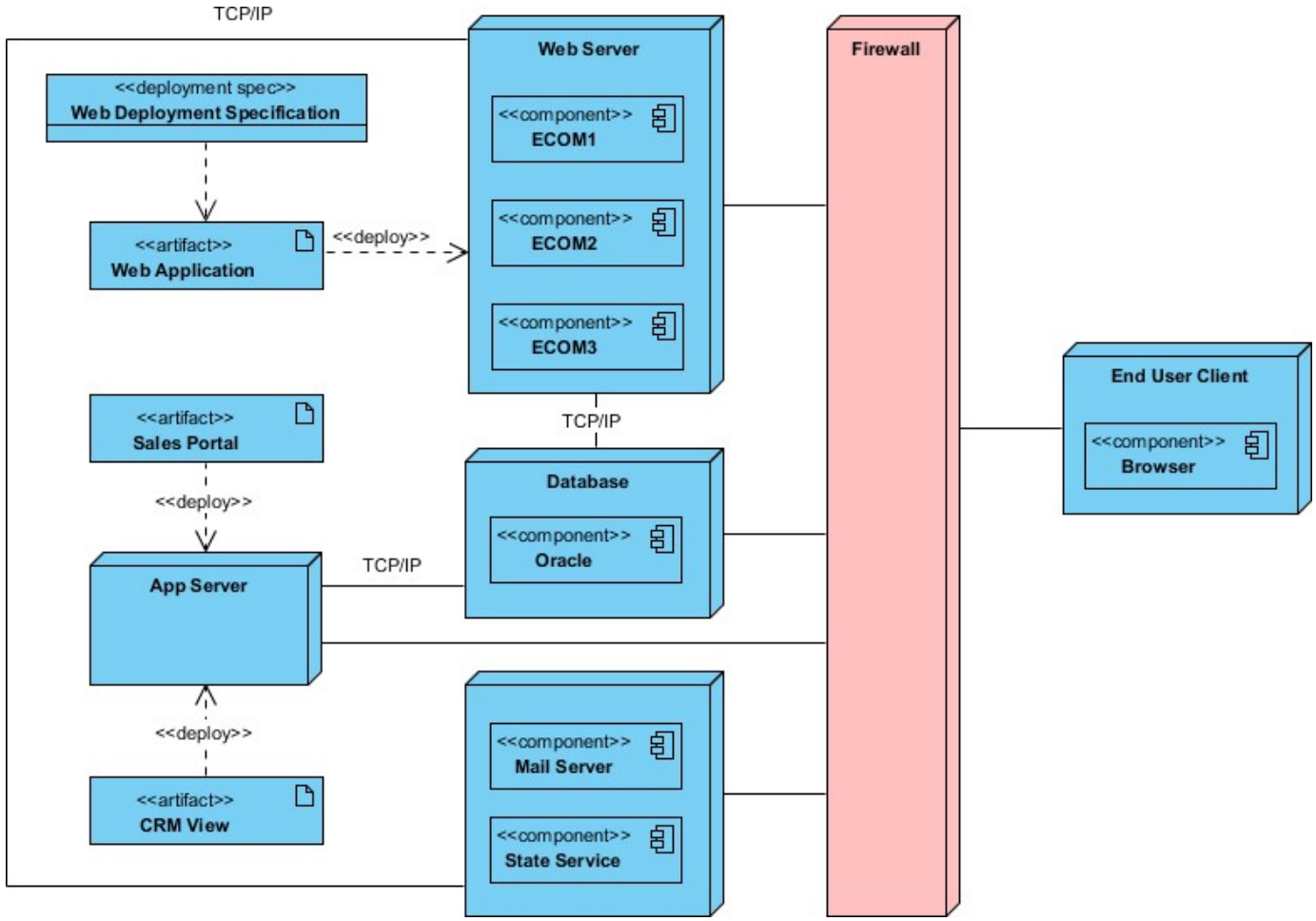
Visão da Implementação – Exemplos



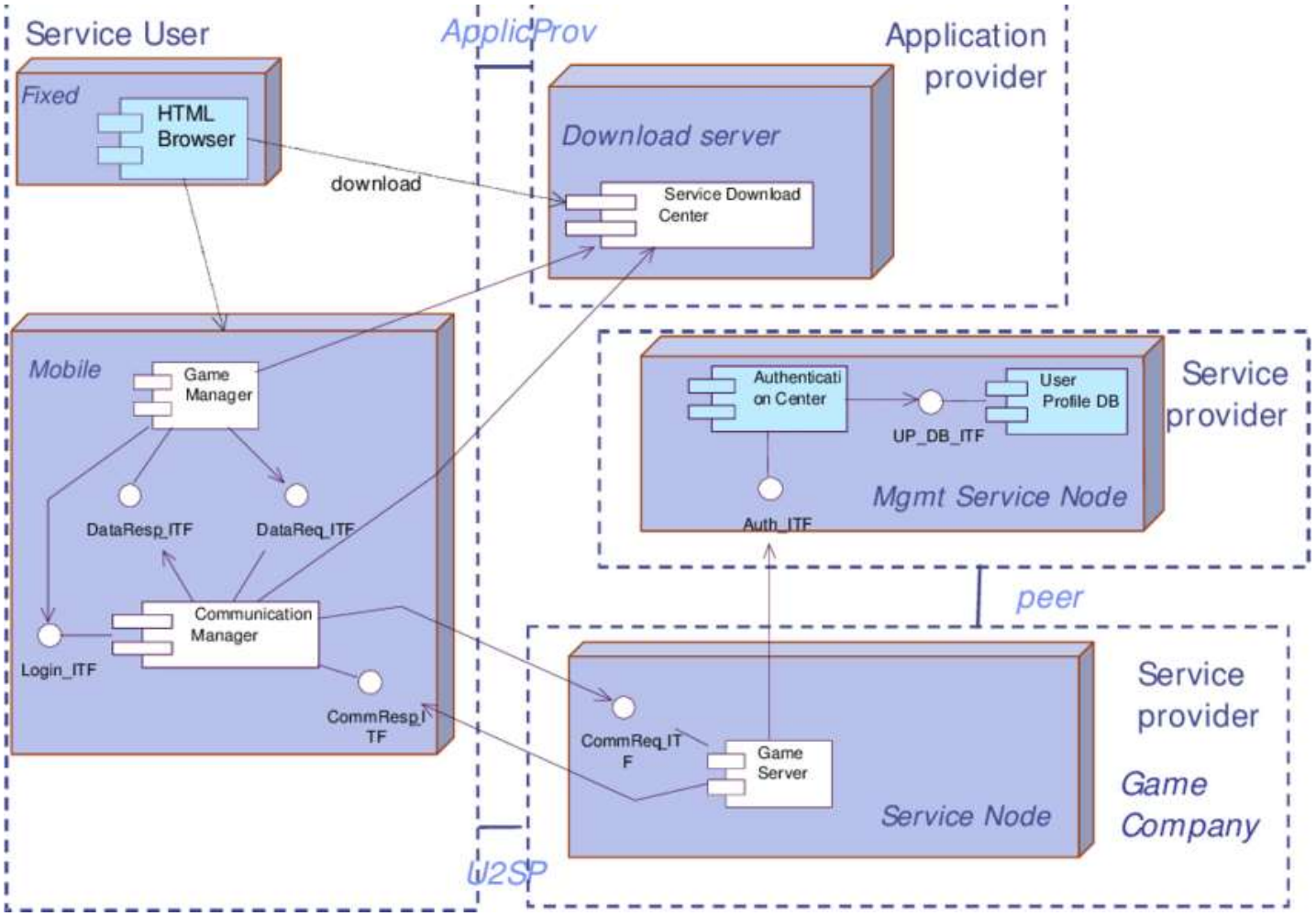
Visão da Implementação – Exemplos



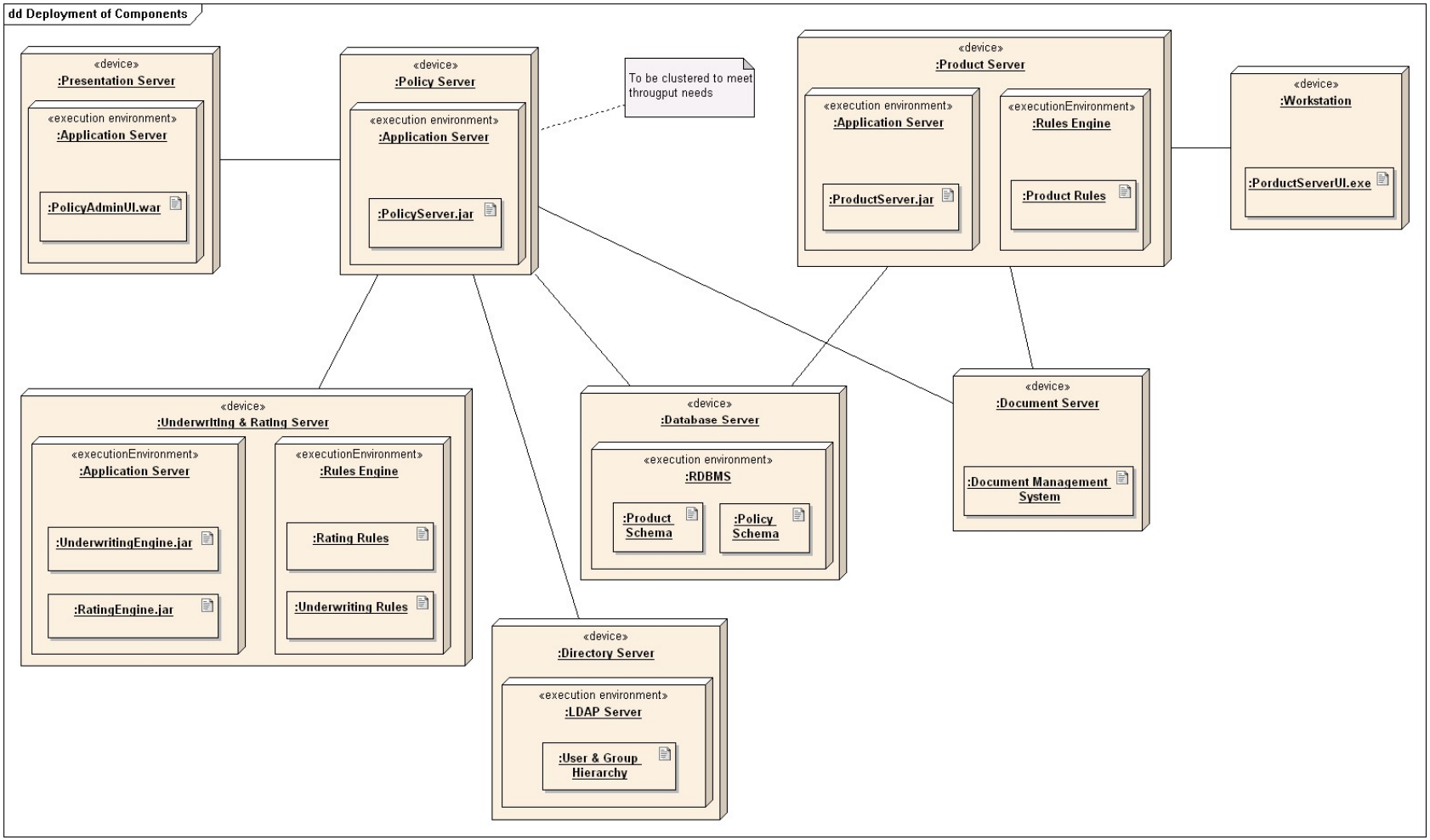
Visão da Implantação - Exemplo



Visão da Implantação - Exemplo



Visão da Implantação - Exemplo



Visão da Implantação - Exemplo