Ciências ULisboa
Faculdade de Ciências da Universidade de Lisboa

2019/2020

# Construção de Sistemas de Software

## Arquitetura de Aplicações Empresariais

### Aplicações Empresariais

- A maior parte dos recursos gastos a desenvolver software é em aplicações empresariais.

*They focus on the display, manipulation, and storage of large amounts of (often complex) data and support the automation of business processes with that data.*

[M.Fowler 2003]

- **Áreas/Exemplos**
  - **CRM** ***C****ustomer* ***R****elationship* ***M****anagement*
  - **SCM** ***S****upply* ***C****hain* ***M****anagement*
  - **SFA** ***S****ales* ***F****orce* ***A****utomation*
  - **ERP** ***E****nterprise* ***R****esource* ***P****lanning*
    - » processamento de ordenados
    - » seguros
    - » acompanhamento de envios

## Aplicações Empresariais: Desafios

Tipos diferentes de software têm diferentes características, que colocam diferentes tipos de desafios. No caso das EAs:

- **Desafios Tecnológicos**
  - Muitos dados persistentes
  - Acessos concorrentes
  - Muito ecrãs de interface com utilizador
  - Interligação com outros sistemas
  - Escalabilidade e Segurança
  - ...

- **Desafios Negócio**
  - Inconsistências entre os conceitos do negócio e os processos
  - Lógica de domínio complexa, difícil de capturar
  - ...



## Aplicações Empresariais: Soluções

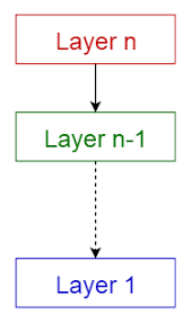
- As soluções evoluíram a par com os avanços tecnológicos

- As diferenças mais importantes são ao nível da **arquitetura do software**, ou seja, nos **elementos** de software em que se baseia a solução e nas **relações** existentes entre eles

- Particularmente relevante são
  - as **camadas** (***layers***) usadas para, do ponto de vista lógico, decompor as funcionalidades da aplicação
  - a forma como estas camadas são, do ponto de vista de processos, combinadas e distribuídas em diferentes **níveis** (***tiers***)

## Arquitetura em Camadas (*Layered Architecture*)

- Consiste na estruturação do código do sistema numa pilha de camadas, cada uma encapsulando um grupo de funcionalidades num particular nível de abstração
  - A camada ***k*** oferece serviços à ***k+1*** servindo-se da ***k-1*** e não tem qualquer dependência de camadas superiores
  - A camada 1, no nível de abstração mais baixo, é a base/núcleo do sistema
  - A camada no topo fornece as funcionalidades de alto nível.






## Arquitetura em Camadas (*Layered Architecture*)

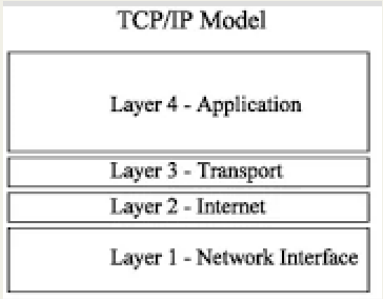
### Exemplos

- Sistemas de comunicação

  TCP/IP tem a *camada física (ethernet, cabo, wireless)*, ligação, internet, transporte e aplicação

- Sistema operativo

  *Hardware*, camada de serviços (sistema de ficheiros, operações I/O, etc), chamadas de sistema, interface de utilizador (linha de comandos, GUI), aplicações de utilizador



## Arquitetura em Camadas (*Layered Architecture*)

Esta forma de estruturação tem diversas **vantagens**:

- **Ajuda a gerir a complexidade**
  - tarefas complicadas concretizadas à custa de tarefas mais simples, endereçadas a diferentes níveis de abstração
- **Mais simples de perceber**
  - responsabilidades de cada camada são mais simples
  - é possível aprender a camada de forma independente do resto do sistema
- **Promove a portabilidade**
  - permite a substituição de uma camada por outra desde que se mantenha o serviço (interface e comportamento)
- **Promove a reutilização**
- **Promove a definição de *standards***
  - as camadas podem ser um bom ponto para padronização (e.g., TCP, IP)

## Arquitetura em Camadas (*Layered Architecture*)

E também algumas **desvantagens**:

- Nem sempre é fácil ou possível estruturar um sistema desta forma
- É difícil estabelecer a granularidade correta das camadas
- As camadas muitas vezes têm "fugas de abstração" (***leaky abstraction***)
- Pode trazer penalizações no desempenho

## Aplicações Empresariais: Arquitetura em Camadas

No caso das aplicações empresariais, as camadas principais são:

- **Camada de Apresentação (*Presentation Layer*)**
- **Camada de Negócio (*Business Logic Layer*)**
- **Camada de Dados (*Data Source Layer*)**


## Aplicações empresariais: Arquitetura em Camadas

No caso das aplicações empresariais, as camadas principais são:

- **Camada de Apresentação (*Presentation Layer*)**
  - responsável pela interface para o mundo através de oferta de serviços e/ou apresentação de informação
    tratamento de pedidos (eventos) GUI do utilizador ou HTTP
- **Camada de Negócio (*Business Logic Layer*)**
  - responsável pela lógica de negócio do sistema
  - i.e., todo o processamento que permite concretizar os pedidos de serviços recebidos na apresentação e calcular eventuais respostas, assim como outros efeitos
- **Camada de Dados (*Data Source Layer*)**
  - responsável pela comunicação com base de dados, sistemas de mensagens, gestores de transações ou outros sistemas
  - i.e., o interface com elementos que nos prestam serviços

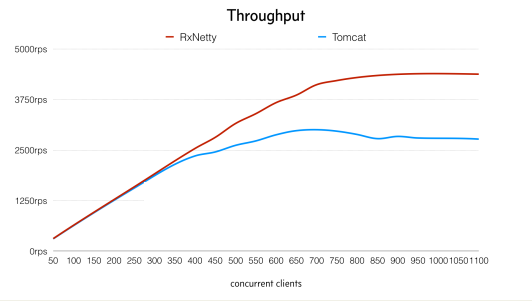
## Aplicações Empresariais: *Tiers*

As soluções evoluíram a par com os avanços tecnológicos.

- **Mainframe**
  - **Monolítico,** ***1-tier***
  - Ou seja, todas as camadas unificadas num único nível
- **PC**
  - ***Client-Server****, 2-tier*
  - Camada de apresentação passa a estar dividida em duas partes, uma que corre do lado do cliente e outra do lado do servidor
- **LAN**
  - ***3-tier***
  - A execução da camada de negócio e os dados deixam de ter de estar co-localizados
  - Permite distribuir a execução da camada de negócio em vários nós
- **Web**
  - ***4-tier***
  - Com uma parte da camada de apresentação responsável pela apresentação web a correr num servidor web e outra parte no cliente (*browser*)

## *Deployment*

- Máquina física vs. máquina virtual
  - JVM
- Correr todo o sistema num servidor ou usar diferentes servidores
  - Servidores Web (em cluster com balanceamento de carga) – apresentação web
  - Servidores Aplicacionais (em cluster com *failover*, etc.) – lógica de negócio
  - Servidores de Base de Dados (em cluster…) – persistência de informação



## *Frameworks*

- **Cliente-Servidor**
  - *Common Object Request Broker Architecture*
  - CORBA/C++, Java, ...

- **Web/Server-side**
  - Java: Java EE, Spring, Hibernate
  - .NET
  - Ruby on Rails
  - Python/Django
  - ...
- Vamos usar o Java EE, o qual define um ***framework* de componentes** estruturado em torno de APIs, e várias implementações abertas para as suas diferentes partes como JBoss, EclipseLink, Hibernate.



## CORBA, J2EE @ Software Architecture L.Bass et al

In the 1980s, the price/performance ratio for personal computers was gradually dovetailing with that of high-end workstations and "servers." This newly available computing power and fast network technology enabled the widespread use of distributed computing.

However, rival computer vendors kept producing competing hardware, operating systems, and network protocols. To an end-user organization, such product differentiation presented problems in distributed computing. Typically, organizations invested in a variety of computing platforms and had difficulty building distributed systems on top of such a heterogeneous environment.

The Object Management Group's Common Object Request Broker Architecture was developed in the early 1990s to counter this problem. The CORBA model provided a standard software platform on which distributed objects could communicate and interact with each other seamlessly and transparently. In this case, an ORB allows objects to publish their interfaces, and it allows client programs to locate them anywhere on the computer network and to request services from them.

However, CORBA was not the only viable distributed object technology for very long. Sun Microsystems soon pushed the Java programming language, which supports remote method invocation (RMI) and so, in effect, builds Java-specific object request broker functionality into every Java Virtual Machine (JVM). Java has the appeal of portability. Once a Java application is developed, its code is portable across all JVMs, which have implementations on most major hardware platforms.

Sun Microsystems did not stop with Java. J2EE was developed in the late 1990s using Java RMI as an underlying communication infrastructure. It became an industry-standard specification for the software community to more easily build distributed object systems using the Java programming language. J2EE soon gathered momentum as software vendors rushed to implement it; Java programmers around the world showed great enthusiasm in developing e-commerce applications in "Internet time" using the J2EE framework. J2EE thus competed directly against CORBA as well as against the proprietary Microsoft technologies.

## *Framework* de Componentes

- Fornece um conjunto de **interações abstratas** que definem os protocolos através dos quais diferentes tipos de **componentes** podem interagir
  - e.g., **JDBC** para acesso a bases de dados baseado em SQL
- Define o ***packaging*** dos componentes de forma a poderem ser instanciados e compostos (*aka* instalados).
  - e.g. *jar, war, ear*
- Fornece um conjunto de funcionalidades *built-in* em componentes consideradas úteis ou processos que encapsulam tarefas comuns (eg., automaticamente instanciar e compor certos componentes)
  - e.g. *servlet containers*



## Sumário

- Aplicações empresariais
  - Características
  - Desafios
- Arquitetura em camadas
- Camadas principais das aplicações empresariais
- Evolução das soluções arquiteturais
  - arquiteturas 1-tier, 2-tier, 3-tier 4-tier
  - vantagens e desvantagens
- *Framework* de componentes