Ambientes de Desenvolvimento Avançados

http://www.dei.isep.ipp.pt/~jtavares/ADAV/ADAV.htm

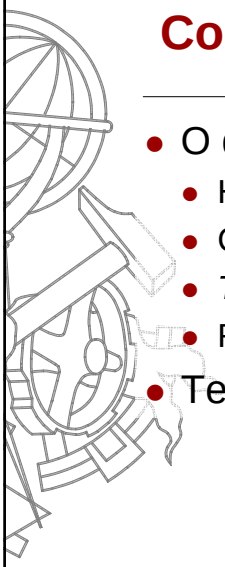
Aula 10

Engenharia Informática

2004/2005

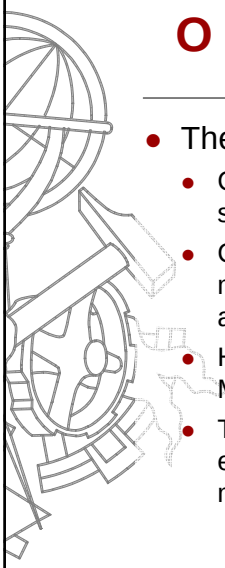
José António Tavares
jrt@isep.ipp.pt

COM:
Uma visão geral



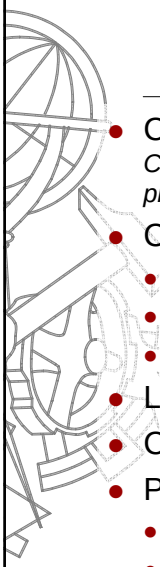
Conteúdo

- O que é o COM
  - História e Background
  - Conceitos Fundamentais
  - *The Component Object Model Defined*
  - Recursos
- Tecnologias COM Existentes



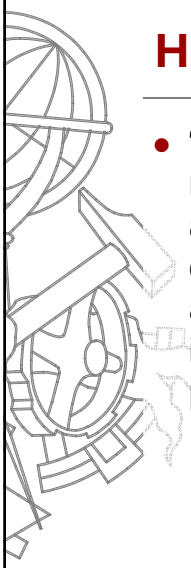
O que é o COM

- The first fundamental wiring model
  - COM is (***was***) Microsoft's foundation on witch all component software on its platform is based;
  - COM is made available on Macintosh by Microsoft and on many other platforms by third parties, such as Software AG and Hewlett-Packard;
  - However, COM has never gained much support beyond the Microsoft Windows platforms;
  - The basic ideas behind COM had quite some influence. For example – the design of the recent CORBA component model.



O que é o COM

- Component software model
  Component def. "…binary, self-contained, reusable, language neutral piece of software"
- COM is a Microsoft solution based on
  - The OO paradigm
  - Binary reuse
  - Interfaces
- Language independent
- Clients bind to objects at runtime
- Provides
  - Integrated services, easy-to-use tools, available applications
  - Market for reusable, off-the-shelf, client and server components



História & *Background*

- "In a sense, Microsoft is taking the easiest route. Instead of proposing a global standard and hoping to port its own system to it, it continually re-engineers its existing application and platform base. Component technology is introduced gradually, gaining leverage from previous successes."

Clemens Szyperski
Component Software, cap. 15

História & *Background*, cont.

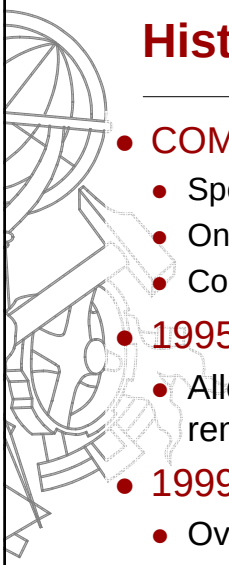
- Microsoft's Component Object Model (COM)
  - The first component model.
    - VBX – non-object-oriented components!
  - 1988 - OO infrastructures that provides reuse based on components
  - OLE2 project
    - OLE - Object Linking & Embedding

4 high-level requirement for a new component architecture

- Component-based
- Based on the OO paradigm
- Language independent
- Interprocess communication

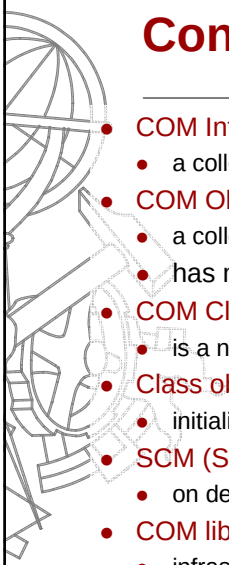
História & *Background*, cont.

- OLE = COM …?!
  - OLE is a technology for linking and embedding document.
  - COM is an architecture for building component-based systems.
  - Therefore - OLE is not the same as COM but, COM was an essential piece of the OLE2 project.
  - The draft COM specification, defined sometime after the completion of the OLE2 project.
  - The specification defines the rules for COM programming.



História & *Background*, cont.

- COM specification
  - Specific memory layout
  - One or more COM interfaces
  - Communication only through interfaces
- 1995 DCOM
  - Allows creating and accessing COM objects on remote location
- 1999
  - Over 150 million systems worldwide



Conceitos Fundamentais

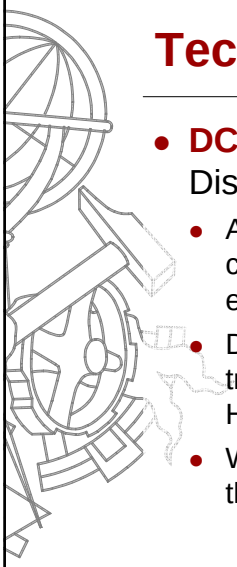
- COM Interface
  - a collection of abstract operations one can perform on an object
- COM Object
  - a collection of vptrs in memory that follow the COM identity laws
  - has methods and state
- COM Class (or coclass)
  - is a named body of code that can be used to produce COM objects
- Class object or Class factory
  - initializes the creation of an object.
- SCM (Service Control Manager) or class loader
  - on demand, load all coclasses
- COM library
  - infrastructure used to support COM on a given platform

The Component Model Defined

- COM interfaces and COM classes
  - Uniquely identified by GUIDs
  - A class may implement several interfaces
  - Several classes may implement the same interface
- COM servers
  - Are either in-process (.dll) or out-of-process (.exe)
  - A server may implement several classes
- The COM library
  - Used by clients to create COM objects

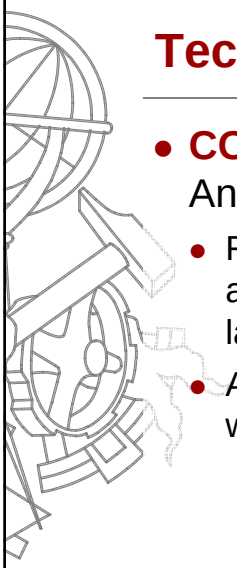
Tecnologias COM Existentes

- DCOM …….. Distributed COM
- COM + …….. Extension of COM
- MSMQ …….. Message Queuing
- MTS …….. Microsoft Transaction Server
- ActiveX Controls
  - Distribution of components over high-latency networks
  - Provide integration of controls into Web browsers



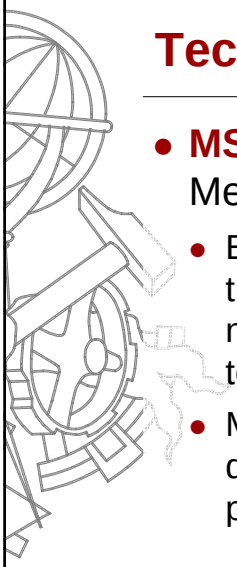
Tecnologias COM Existentes

- **DCOM**
  Distributed Component Object Model.
  - A protocol that enables software components to communicate directly over a reliable, secure and efficient manner.
  - DCOM is designed for use across multiple network transports, including Internet Protocols such as HTTP.
  - Work with Java applets & ActiveX components through its use of COM.



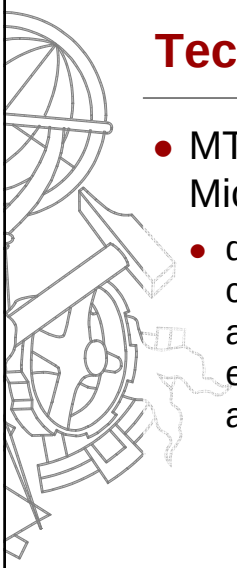
Tecnologias COM Existentes

- **COM +**
  An extension of COM.
  - For making it easier for developers to create and use software components in any language, using any tool.
  - Applications currently using COM technology will work in the COM+ environment.



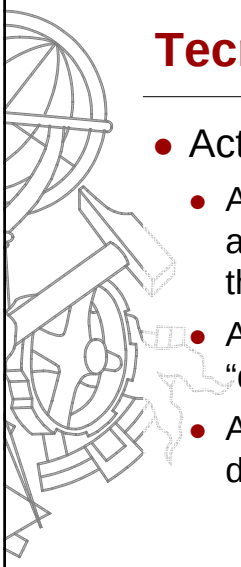
Tecnologias COM Existentes

- **MSMQ**
  Message Queuing technology
  - Enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline.
  - MSMQ provides guarantied message delivery, efficient routing, security, and priority-based messaging.



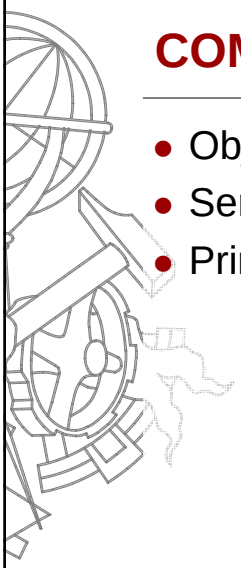
Tecnologias COM Existentes

- MTS
  Microsoft Transaction Server:
  - delivers; transactions, scalability services, connection management, and point-and-click administration providing developers with an easy way to build and deploy scalable server applications for business and the Internet.



Tecnologias COM Existentes

- ActiveX
  - A marketing name for a set of technologies and services, all based on COM (the model, the “ORB”, and the services).
  - ActiveX Controls are COM components with “design-time” UI.
  - ActiveX is Self-registering, Optimized for download and execute.

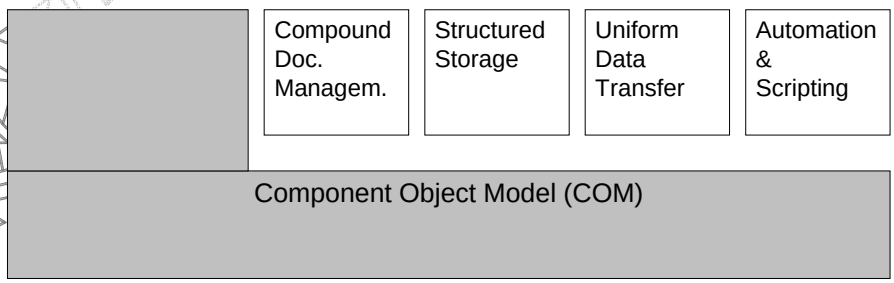


COM/OLE - Conteúdo

- Objecto COM
- Servidor COM
- Principais Serviços dos Objectos COM

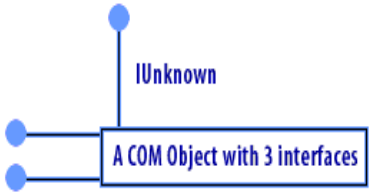
COM/OLE

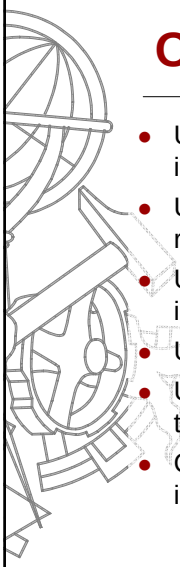
- A imagem seguinte representa os componentes principais da arquitectura do COM/OLE.
- Regra geral tudo o que não faça parte dos documentos compostos OLE pode ser visto como serviços COM.



Object-Based Technology

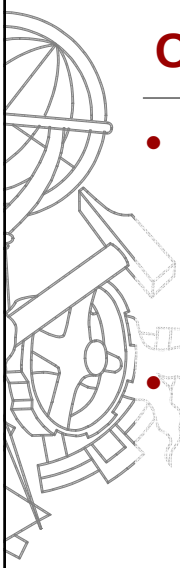
- COM specifies the behavior of objects and how to integrate (use) objects
- A COM object can be thought of as a container for one or more COM interfaces
- To denote a COM object and its interfaces, COM uses a box with connected interface diagrams:





Objecto COM

- Um objecto COM é um componente que suporta um ou mais interfaces definidos pela classe do objecto;
- Um interface COM consiste num grupo predefinido de funções relacionadas;
- Uma classe COM implementa um ou mais interfaces e é identificada por um número único de 128 bits;
- Um objecto COM é uma instância de uma classe;
- Um objecto providencia as implementações das funções de todos os interfaces da sua classe;
- Os clientes utilizam objectos COM através de ponteiros para interfaces (nunca acede directamente ao objecto).

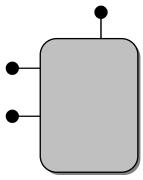


Objecto COM

- Todos os objectos COM têm que implementar o interface **IUnknown**, que permite que o cliente controle o tempo de vida de um objecto. Este interface também permite que os clientes façam inquéritos sobre os interfaces suportados pelo objecto.
- Os interfaces também têm o seu identificador (número único de 128 bits).

Objecto COM Simples

- A COM object has methods and state
  - Language independent
- All Access is through the interface
  - Supports multiple interfaces
  - Each interface has a GUID



Servidor COM

- Um servidor COM é um 'pedaço' de *software* (EXE ou DLL) que contém uma ou mais classes de objectos cada uma com o seu identificador único (CLSID).
- Quando um cliente requisita um objecto de uma determinada classe (CLSID), o COM carrega o código do servidor e pede que este crie um objecto da classe. O servidor deve implementar aquilo que se designa de ***class factory*** para criar objectos.
- Quando um objecto é criado o COM retorna um ponteiro para o seu interface primário.
- O servidor COM não é um objecto COM.
- Um servidor COM implementa o necessário de forma a disponibilizar objectos aos seus clientes.

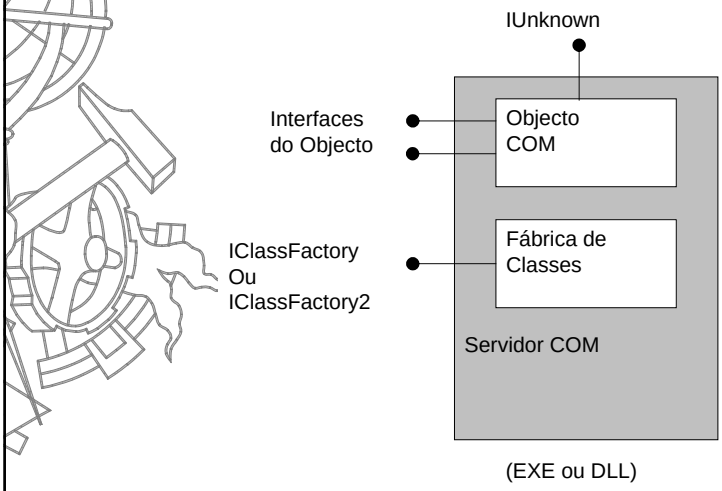
Servidor COM

- O servidor COM deve:
  - Implementar um interface '**class factory**'.
    O servidor deve implementar uma 'fábrica' de classes para cada `CLSID` suportado através do interface `IClassFactory`. Se a classe suportar licenciamento então deve também implementar o interface `IClassFactory2`. Neste caso o interface apenas cria o objecto se existe um ficheiro de licença ou é fornecida uma chave de licença.
  - Registar as classes que suporta.
    Consiste basicamente em registar os `CLSID` para cada classe que é suportada no registry do Windows. Para cada `CLSID` é colocado no registry o path para o servidor que a implementa (DLL ou EXE).

Servidor COM

- O servidor COM deve:
  - Inicializar a biblioteca COM.
    Consiste em chamar a função `CoInitialize` da biblioteca COM (OLE32.DLL). Esta dll contem serviços base de suporte ao COM. Todas as funções desta biblioteca começam por Co.
  - Verificar que a biblioteca é de uma versão compatível.
    Através da função `CoBuildVersion`.
  - Implementar um mecanismo para se descarregar da memória.
    Deve ser um método automático que acontece quando não existem clientes a utilizar objectos do servidor.
  - Finalizar a biblioteca COM.
    Através da função `CoUninitialize`.

Servidor COM

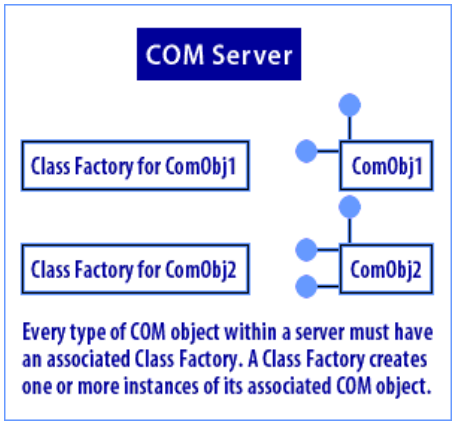


Class Factory

- *Class factory* is one kind of class object (the most common kind)
- A class factory implements an interface called `IClassFactory`
- `IClassFactory` defines a method that creates an instance of a COM object

Class Factories

- One dedicated class factory for each type of COM object supported
- If a server implements COM objects A and B, it must implement two class factories: one for A and one for B



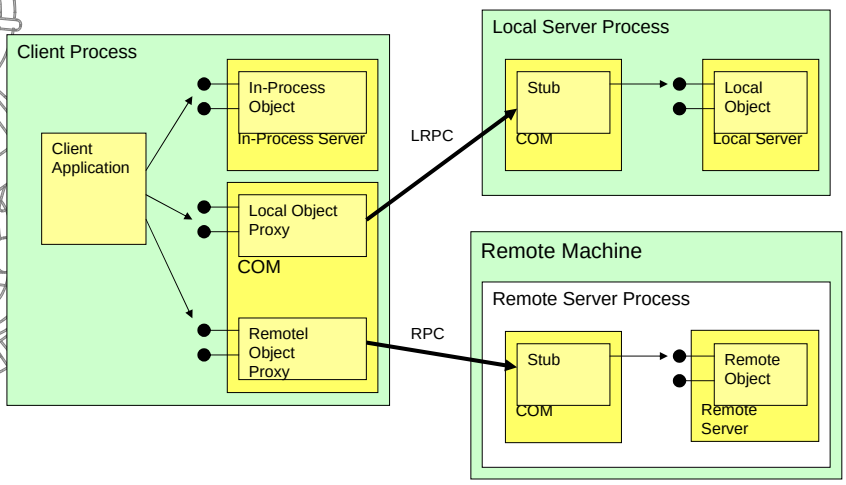
Criação de um Objecto COM

- A client asks COM for the class factory that knows how to create a specific COM object
- COM asks the appropriate COM server for the requested class factory
- The server returns a pointer to the class factory's `IClassFactory` interface to COM
- COM returns it to the client
- The client uses the `CreateInstance` method in `IClassFactory` to create instances of the COM object

Tipos de Servidores COM

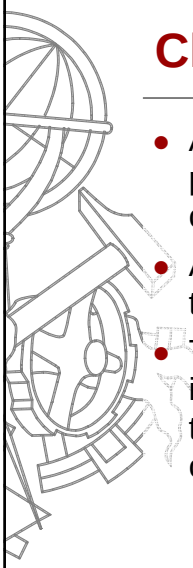
- **In-process Servers**
  - Executam no mesmo espaço de endereçamento que os clientes.
  - São implementados como DLL que são carregadas no processo cliente.
- **Local Servers**
  - Executam num processo diferente do cliente mas na mesma máquina (e sistema operativo).
  - Os clientes usam LRPC para comunicar com o servidor local.
  - O servidor local executado é um EXE.
- **Remote Servers**
  - Executam num processo separado numa máquina remota e possivelmente num sistema operativo diferente.
  - Os clientes usam RPC para comunicar com o servidor.

Exemplo



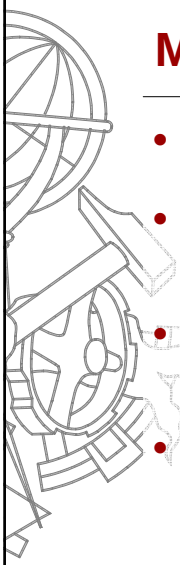
Tipos de Servidores COM

- Os clientes podem comunicar com os servidores COM independentemente do local onde estes correm.
- Do ponto de vista dos clientes todos os objectos são acedidos através de ponteiros para os seus interfaces e um ponteiro só pode ter existência e validade no espaço de endereçamento de um processo.



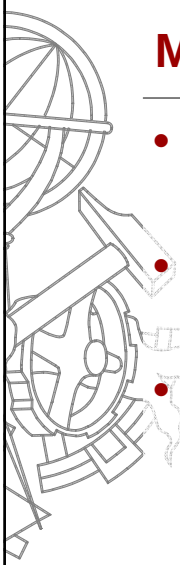
Clientes e Servidores COM

- A COM client is whatever code or object gets a pointer to a COM server and uses its services by calling the methods of its interface(s)
- A COM server is any object that provides services to clients
- These services are in the form of COM interface implementations that can be called by any client that is able to get a pointer to one of the interfaces on the server object



Mesma Máquina

- For clients and servers on the same machine, the **`CLSID`** of the server is all the client ever needs
- On each machine, COM maintains a database (using the system registry on Windows and Macintosh) of all the **`CLSIDs`** for the servers installed on the system
- This is a mapping between each **`CLSID`** and the location of the DLL or EXE that houses the code for that **`CLSID`**
- COM consults this database whenever a client wants to create an instance of a COM class and use its services, so the client never needs to know the absolute location



Máquinas diferentes

- For distributed systems, COM provides registry entries that allow a remote server to register itself for use by a local client
- Applications need know only a server's **`CLSID`**, because they can rely on the registry to locate the server
- However, COM allows clients to override registry entries and to specify server locations, to take full advantage of the network

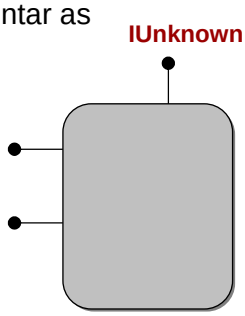
Localização transparente

- Abstraction, or hiding, of the location of the server
- Having obtained a pointer to a COM interface, a client simply makes calls into the interface
- To the client, there is no difference between calling a method in a server on the same machine or calling a machine across the network
- The COM runtime handles the details involved with making the connection to the server and calls to servers in different processes

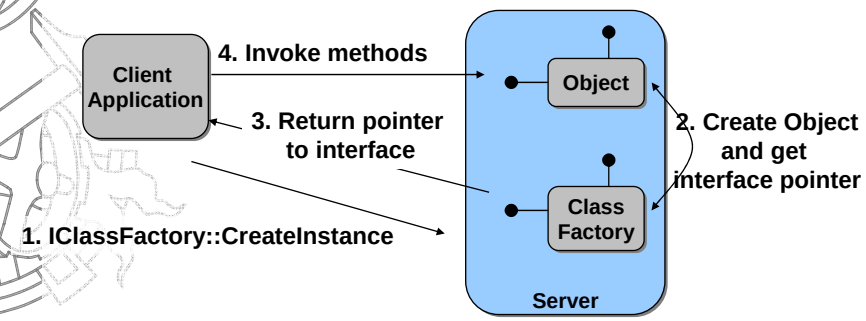
Principais Serviços dos Objectos COM

Interface `IUnknown`

- O interface `IUnknown` é utilizado para 'negociações' de interfaces em tempo de execução, gestão de vida do objecto e agregação.
- Qualquer Interface COM tem que implementar as funções membro do interface `IUnknown`:
  - `AddRef`
  - `QueryInterface`
  - `Release`



Criação de um objecto local usando a *class factory*



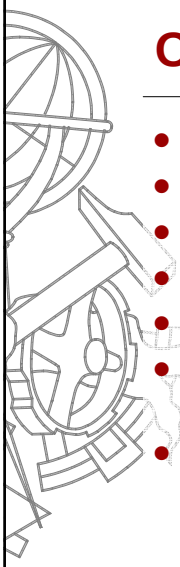
Principais Serviços dos Objectos COM

Interface IClassFactory ou IClassFactory

- Cada classe COM tem que implementar uma '*class factory*' que um servidor COM possa invocar para criar instâncias dessa classe.
- Uma 'class factory' consiste numa implementação do interface IClassFactory ou IClassFactory2.

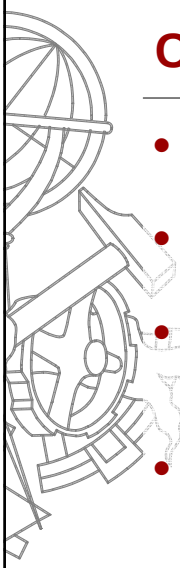
Membros de IClassFactory:
- *QueryInterface*
- *AddRef*
- *Release*
- CreateInstance
- LockServer

Membros de IClassFactory2:
- *QueryInterface*
- *AddRef*
- *Release*
- *CreateInstance*
- *LockServer*
- GetLicInfo
- RequestLicKey
- CreateInstanceLic



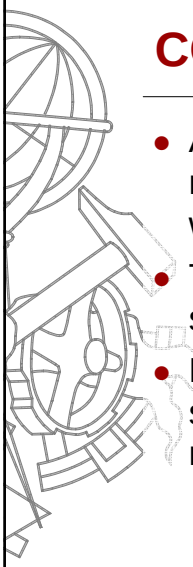
Outras facilidades

- Processes, threads and apartments
- Containment, delegation, aggregation
- Persistent objects
- Asynchronous calls
- Canceling asynchronous and synchronous calls
- Security – authentication and authorization for activation and call
- …



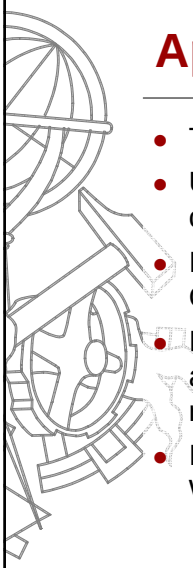
COM Apartment

- COM objects are actually created in the context of a COM *apartment*
- A COM apartment defines the threading context of an executing COM object
- A *single-threaded apartment* (STA) allows only one thread to access interface methods within the object
- This is always the same thread



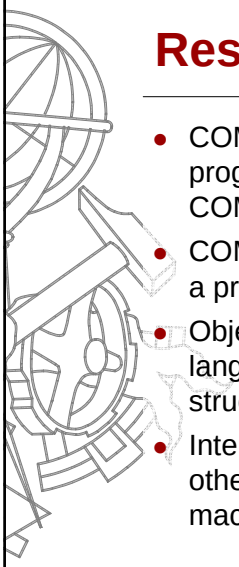
COM Apartment

- A *multithreaded apartment* (MTA) allows multiple threads to access interface methods within the same object
- This access can, and often does, occur simultaneously
- For example, thread A and thread B can simultaneously access the same or different methods within the object



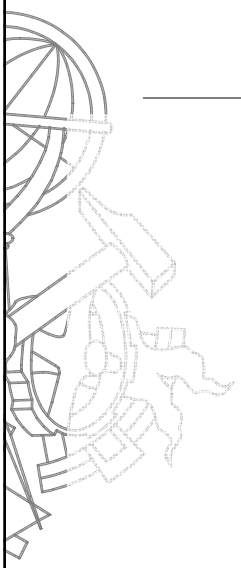
Apartment Objects

- There aren't any
- Unlike COM interfaces and objects, which are visible in code, there is no apartment object
- Instead, the type of apartment dictates how to code COM interface methods
- In an STA, because only one specific thread can access an instance of a COM object, no special coding is required to guard data and methods
- In an MTA, data and methods are often guarded using Win32 synchronization objects



Resumo

- COM specifies an object (or component) model, and programming and compiler requirements, that enable COM objects to interact with other COM objects
- COM is a *binary* standard—a standard that applies after a program has been translated to binary machine code
- Objects can be written in different languages, including languages that don't have "objects", and may be structurally quite dissimilar
- Interacting objects can be within a single process, in other processes on the same machine, or on remote machines



Questões

?