Arquitetura do DCOM

Alexandre Ricardo Nardi
nardi@ime.usp.br

Objetivo

- Apresentar uma visão geral da infra-estrutura e funcionamento do COM/DCOM, sem entrar em detalhes sobre os serviços oferecidos, como de transações, chamadas assíncronas,...
- Estabelecer parâmetros que possibilitem comparação com o CORBA, assunto do seminário “COM ou CORBA ?”

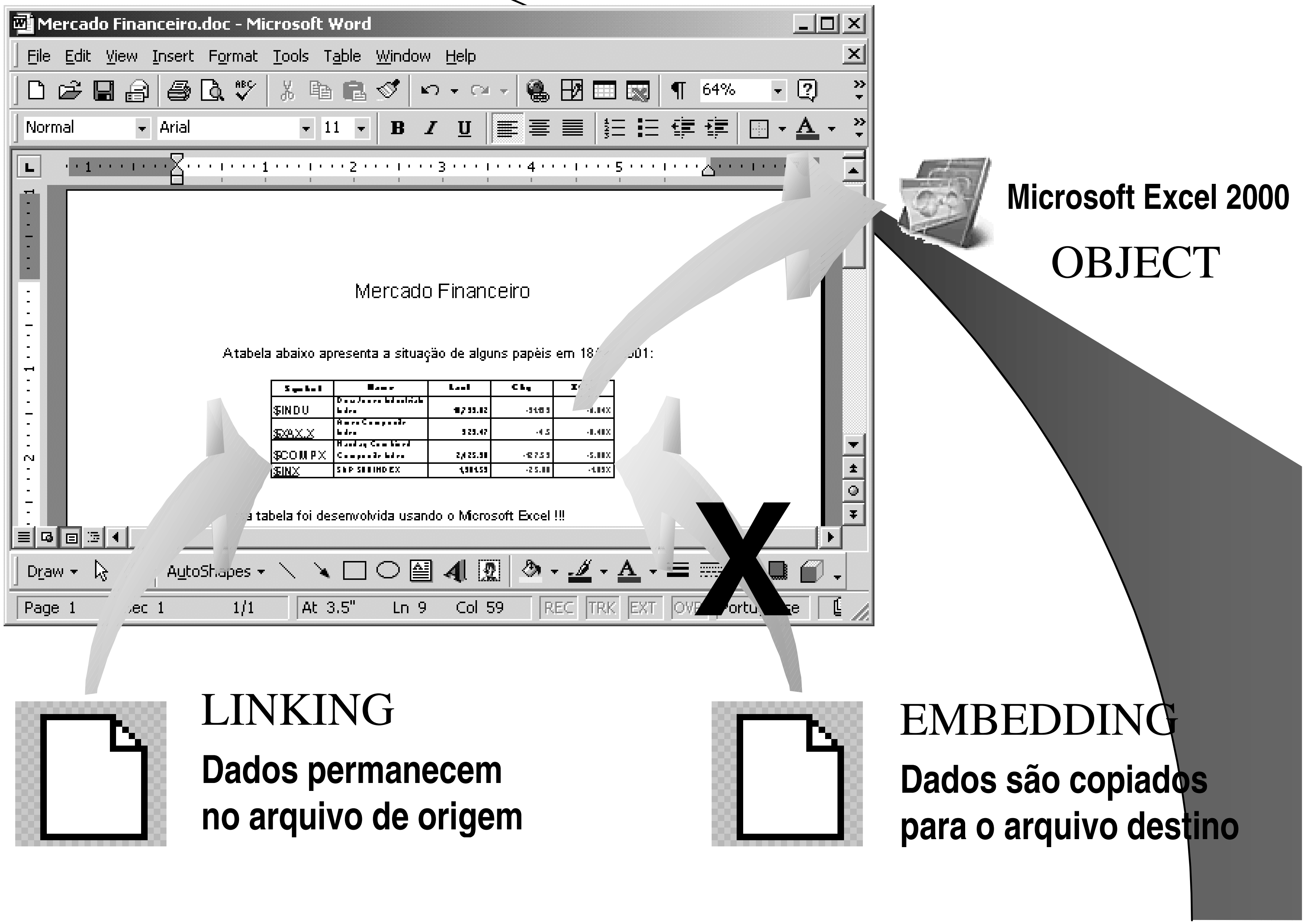
Roteiro

- Histórico
- Objetos X Componentes
- Fundamentos do COM/DCOM
  - Interfaces
  - Acessando um objeto
  - Tipos de servidores
  - Proxies e Stubs
  - Marshaling
  - Criação de objetos
  - Destruição de objetos
- Referências

Histórico

- 1988: Concepção do COM
  - Influenciado por trabalhos prévios em Smalltalk, C++ e OSF Distributed Computing Environment (DCE)
- 1993: Primeira versão do COM, como parte do OLE 2.0 SDK
  - Apenas comunicação local
  - 16-bit/single-threaded
- 1994: Lançamento do Windows NT 3.5
  - Suporte para 32-bits + IDL
- 1995: Windows NT 3.51/Windows 95
  - Pequenas melhorias, como suporte para ambiente multi-thread
- 1996: Windows NT 4.0
  - DCE-based Distributed COM (DCOM) protocol
  - Melhorias em segurança e threading
- 1996: Microsoft Transaction Server (MTS) 1.0
  - Component-based Distributed Application Framework
- 1998: Windows NT 4.0 Option Pack
  - Microsoft Transaction Server 2.0
  - Microsoft Message Queue (MSMQ) 1.0
  - Internet Information Server 4.0
- 2000: Windows 2000
  - COM+, com a incorporação de diversos serviços

Histórico


Mercado Financeiro.doc - Microsoft Word
Mercado Financeiro
Microsoft Excel 2000
OBJECT
X
LINKING
Dados permanecem
no arquivo de origem
EMBEDDING
Dados são copiados
para o arquivo destino

Histórico

- OLE
  - Visual Editing
- OLE 2.0
  - Automation
- ActiveX
  - ActiveX Controls
  - ActiveX Documents
  - ActiveX DLL e EXE
- Windows DNA (Distributed interNet Applications Architecture)
  - Práticas de programação recomendadas
- Microsoft .NET
  - Em desenvolvimento

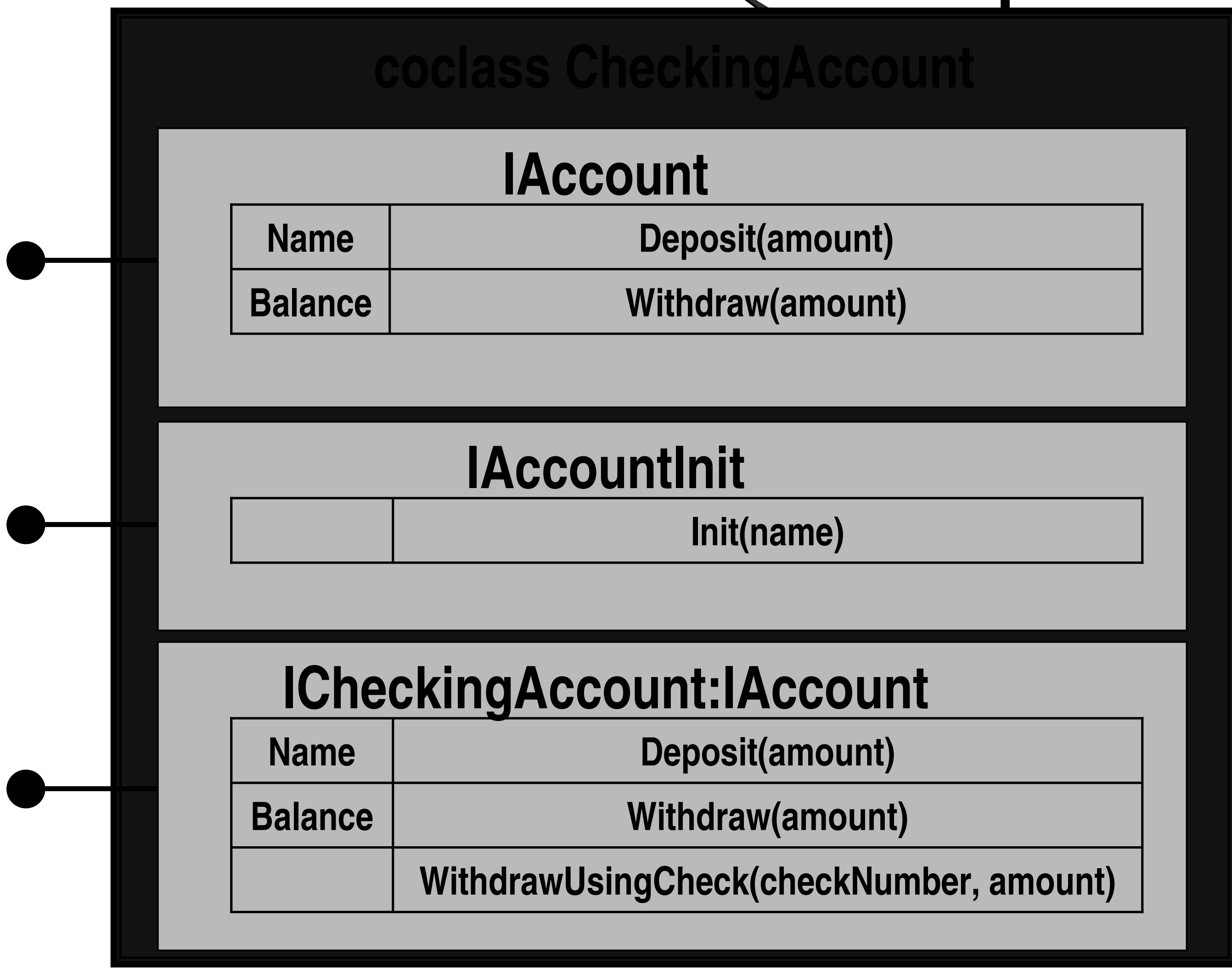
Objetos X Componentes

- Objetos
  - Prioridade para encapsulamento e reutilização de implementação através de herança
  - Interfaces de objetos representam um contrato
- Componentes
  - Prioridade para conectividade (pluggability)
  - É possível implementar um componente sem usar objetos!
  - Encapsulamento é conseqüência
  - Herança deveria ser resultado de bom design

Fundamentos do COM/DCOM Interfaces

- São contratos que descrevem as funcionalidades disponibilizadas pelos objetos do sistema
- Definidas através da linguagem IDL (da Microsoft) – Interface Definition Language
- Por vezes substituído por Type Libraries, arquivos compilados (binários) que contém uma descrição padronizada de um objeto COM. Muito utilizado por clientes VB – facilita a codificação, eliminando a necessidade de conhecer a estrutura da IDL. Entretanto, muitas vezes é necessário utilizar IDL
- COM suporta herança (simples) de interfaces, mas não de implementação

Fundamentos do COM/DCOM Interfaces – Um exemplo



Fundamentos do COM/DCOM
Interfaces - IDL

```
import "oaidl.idl";
import "ocidl.idl";

  [   object,
      uuid(B5F3E2FE-B376-11D1-BB1E-00207812E629),
      oleautomation,
      helpstring("IAccount Interface"),
      pointer_default(unique)
  ]
  interface IAccount : IUnknown
  {   [propget, helpstring("property Balance")]
      HRESULT Balance([out, retval] double *pVal);

      [propget, helpstring("property Name")]
      HRESULT Name([out, retval] BSTR *pVal);

      [helpstring("method Deposit")]
      HRESULT Deposit([in] double amount);
      [helpstring("method Withdraw")]
      HRESULT Withdraw([in] double amount);
  };

  [   object,
      uuid(B5F3E2FF-B376-11D1-BB1E-00207812E629),
      oleautomation,
      helpstring("IAccountInit Interface"),
      pointer_default(unique)
  ]
  interface IAccountInit : IUnknown
  {   const int E_INVALID_NAME = 0x80040200;

       [helpstring("method Init")]
      HRESULT Init([in] BSTR name);
  };
```

Fundamentos do COM/DCOM
Interfaces - IDL

```
[   object,
    uuid(B5F3E2FD-B376-11D1-BB1E-00207812E629),
    oleautomation,
    helpstring("ICheckingAccount Interface"),
    pointer_default(unique)
]
interface ICheckingAccount : IAccount
{   [helpstring("method WithdrawUsingCheck")]
    HRESULT WithdrawUsingCheck(
            [in] long checkNumber, [in] double
    amount);
};
```

```
[   uuid(B5F3E2F0-B376-11D1-BB1E-00207812E629),
    version(1.0),
    helpstring("ComCppServer 1.0 Type Library")
]
library COMCPPSERVERLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    [   uuid(B5F3E300-B376-11D1-BB1E-00207812E629),
        helpstring("Ch3CheckingAccount Class")
    ]
    coclass CheckingAccount
    {   interface IAccount;
        interface IAccountInit;
        [default] interface ICheckingAccount;
    };
};
```

Fundamentos do COM/DCOM Interface IUnknown

- Deve ser implementada por todo objeto COM/DCOM
- Métodos:
  - QueryInterface: retorna um ponteiro para uma interface
  - AddRef: quando um cliente recebe um interface pointer de outro cliente, o primeiro deve chamar este método
  - Release: chamado por todo cliente, quando não precisar mais do objeto

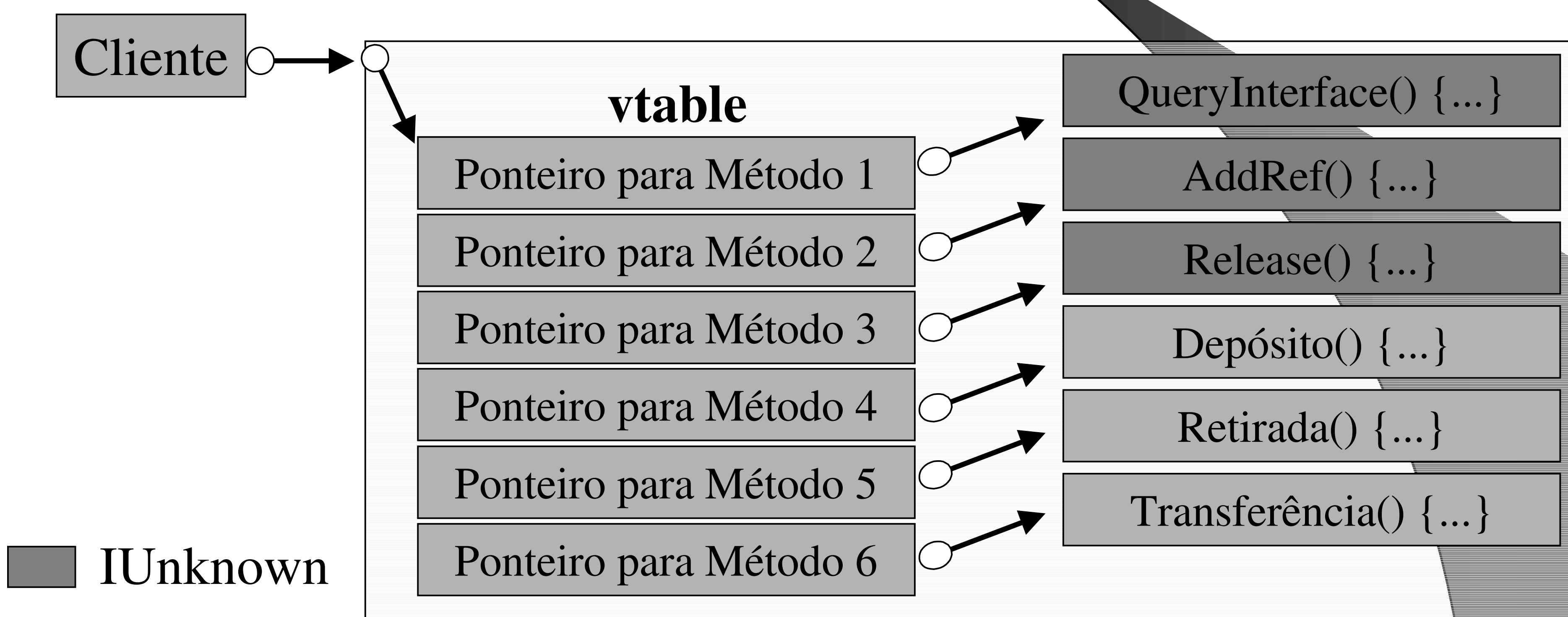
Fundamentos do COM/DCOM Interface IUnknown - Exemplo

```
BOOL MyCheckingAccount::SetInterfaces(
   IUnknown* pIUnknown)
{
   // Reset all interface pointers.
   Reset();
   HRESULT hr;
   void*   pInterface;

   // Query the IUnknown interface to get the
   // ICheckingAccount interface.
   hr = pIUnknown->QueryInterface(
      IID_ICheckingAccount, &pInterface);
   m_pICheckingAccount =
      ( ICheckingAccount*) pInterface;

   // Query the ICheckingAccount interface to get the
   // IAccount interface.
   hr = m_pICheckingAccount->QueryInterface(
      IID_IAccount, &pInterface);
   m_pIAccount = (IAccount*) pInterface;

   // Query the IAccount interface to get the
   // IAccountInit interface.
   hr = m_pIAccount->QueryInterface(
      IID_IAccountInit, &pInterface);
   m_pIAccountInit = (IAccountInit*)pInterface;

   return TRUE;
}
```

Fundamentos do COM/DCOM Tipos de Interfaces

- Custom Interfaces
  - COM IDL suporta o tratamento de tipos simples (*int*, *long*,...) e complexos (*arrays, structs*,...)
  - Nem todo ambiente de desenvolvimento suporta tipos complexos (VB, VJ++)
  - Custom interfaces são eficientes, tendo mapeamento direto em *vtables*

Fundamentos do COM/DCOM
Acessando um Objeto – Custom Interface

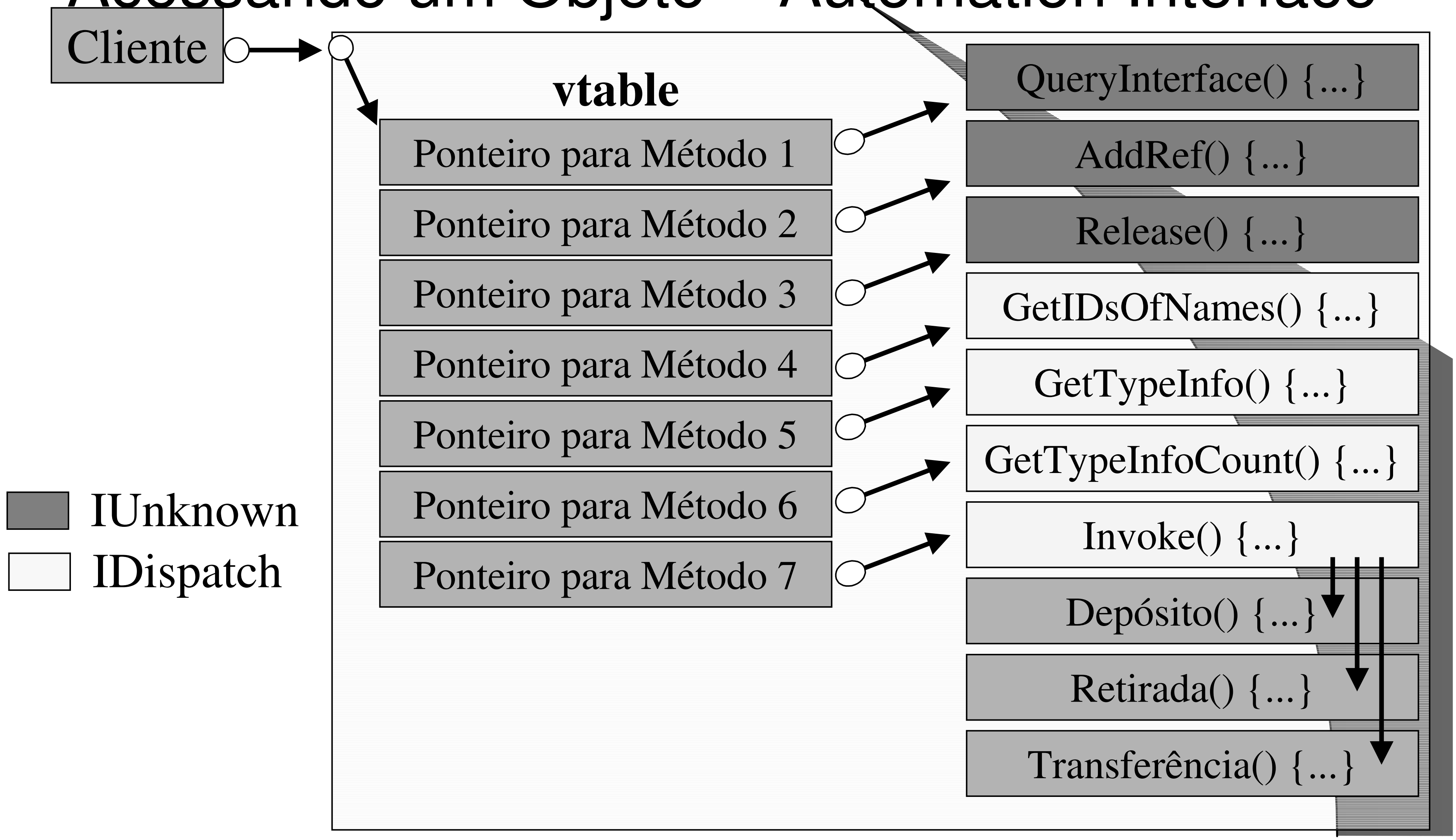


Fundamentos do COM/DCOM
Tipos de Interfaces

- Automation Interfaces
  - Implementação da interface IDispatch para suporte a invocação dinâmica de métodos, comum em ambientes interpretados (VBScript, JScript)
  - Suporte para *automation types*
  - Eficiência pelo menos 10x menor do que uso de custom interfaces

Fundamentos do COM/DCOM

Acessando um Objeto – Automation Interface

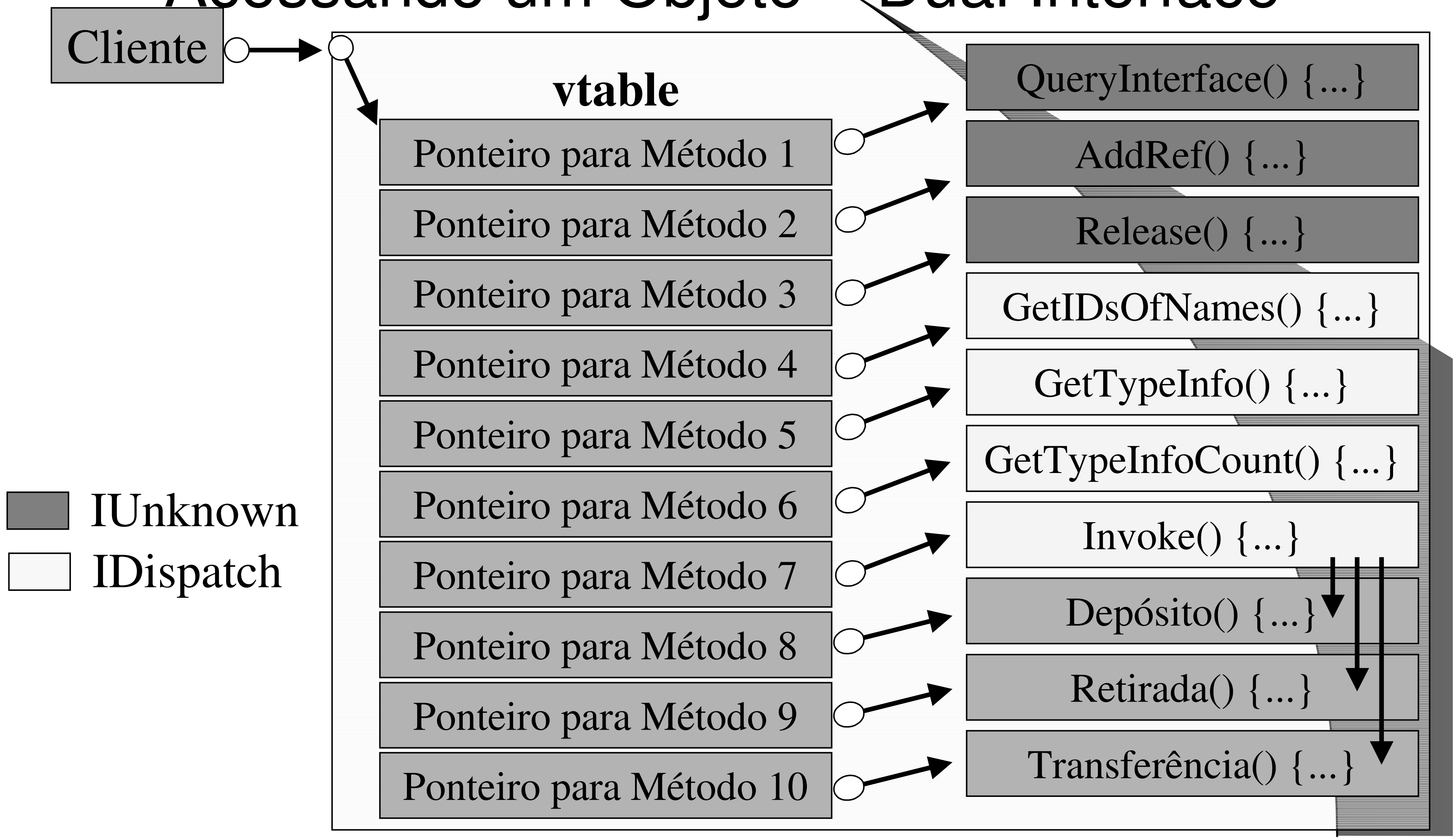


Fundamentos do COM/DCOM Tipos de Interfaces

- Dual Interfaces
  - Suporte para interfaces convencionais e de automação
  - Limitação: deve utilizar apenas *automation types*

Fundamentos do COM/DCOM

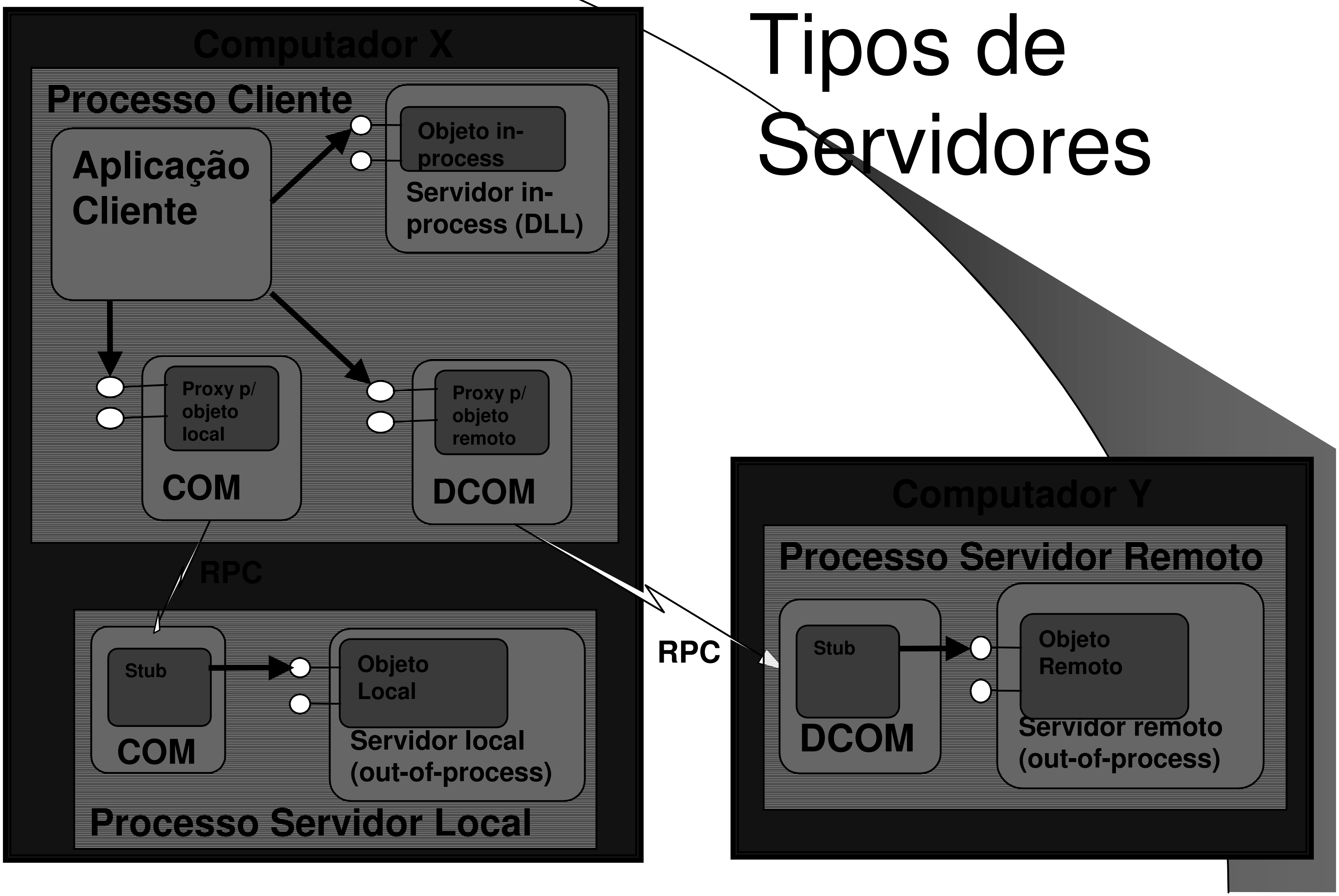
Acessando um Objeto – Dual Interface



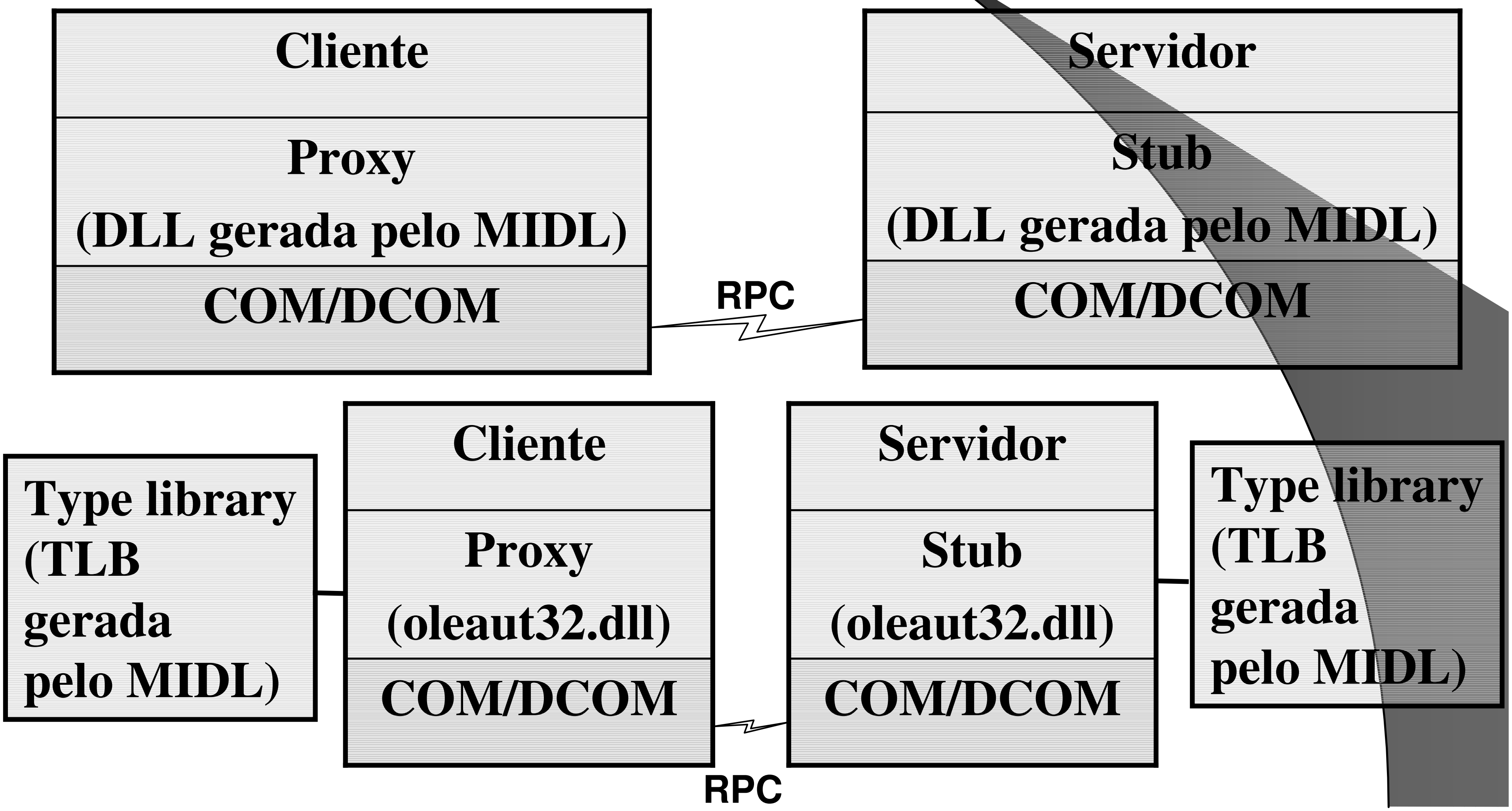
Fundamentos do COM/DCOM
Tipos de Dados - Automação

COM IDL Type	Descrição
short	inteiro de 16 bits com sinal
long	inteiro de 32 bits com sinal
float	número em ponto flutuante com 32 bits
double	número em ponto flutuante com 64 bits
BSTR	array com tamanho pré-fixado de caracteres de 2 bytes
VARIANT_BOOL	short integer used to indicate true or false
DATE	double representando datas
IUnknown*	ponteiro genérico para interface
IDispatch*	ponteiro para interface de automação
VARIANT	union de todos os tipos de automação
SAFEARRAY	para gerenciar arrays multi-dimensionais de tamanho variável

Fundamentos do COM/DCOM
Tipos de Servidores



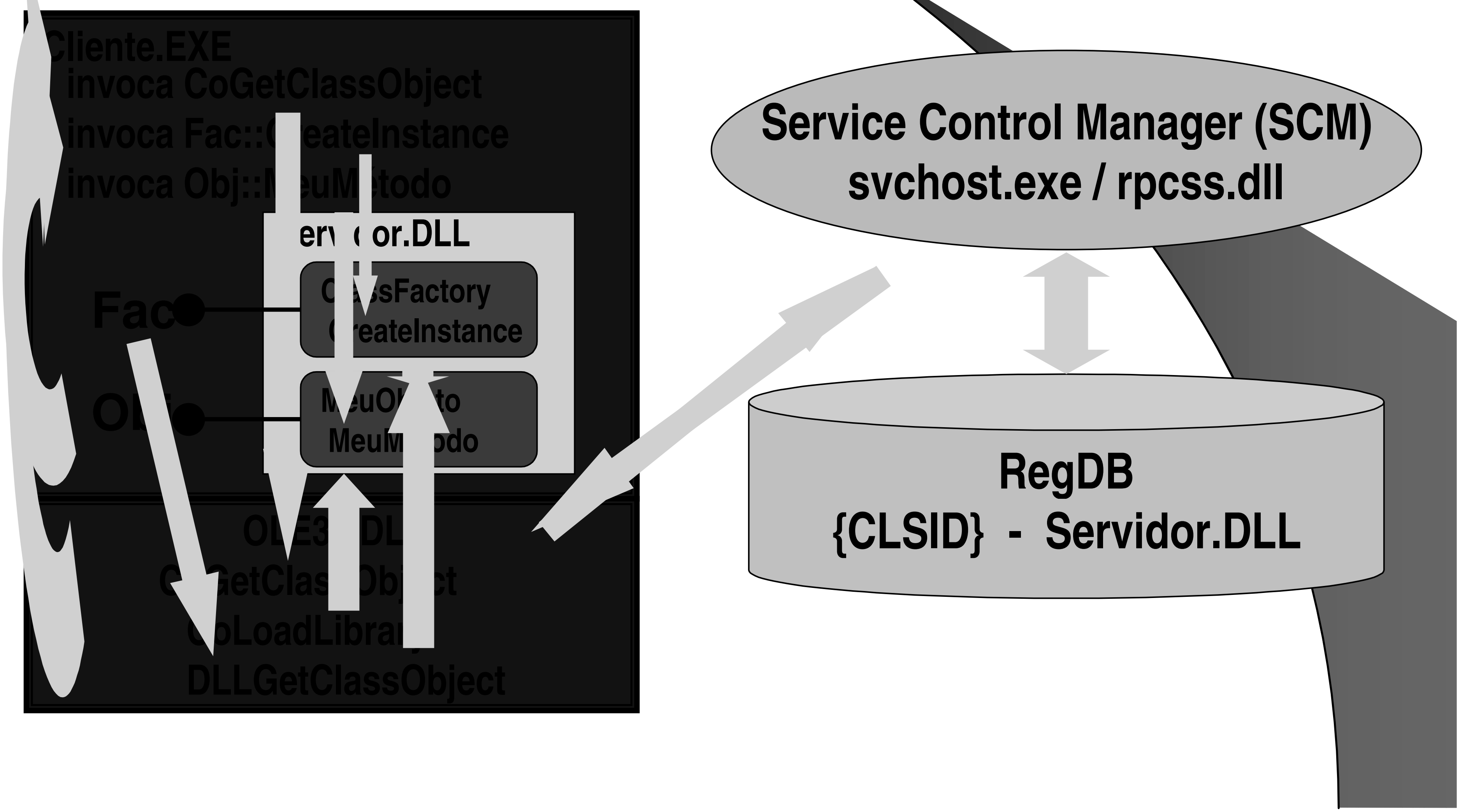
Fundamentos do COM/DCOM
Proxies e Stubs



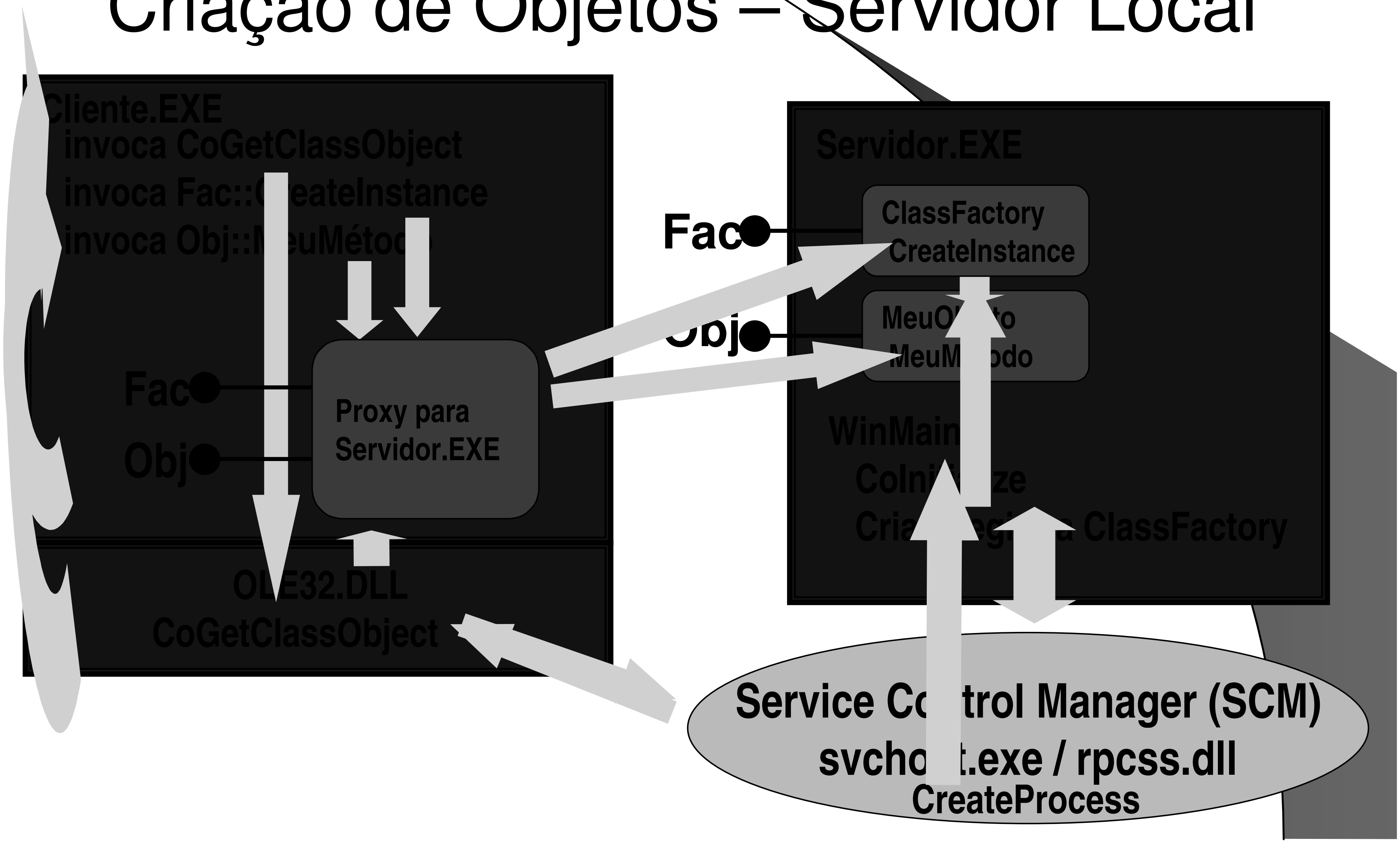
Fundamentos do COM/DCOM Marshaling

Tipo de interface cliente	Tipo de Marshaling
Custom interface	Proxy/stub (DLL) deve ser registrada em cada computador cliente
Custom interface como [oleautomation]	Registrar type library ou registrar proxy/stub (DLL) em cada cliente
Automation interface	IDispatch já está instalada (nada a fazer)
Dual interface	Registrar type library ou registrar proxy/stub (DLL) em cada cliente Nos clientes usando apenas IDispatch, nada a fazer

Fundamentos do COM/DCOM
Criação de Objetos – Servidor In-Process

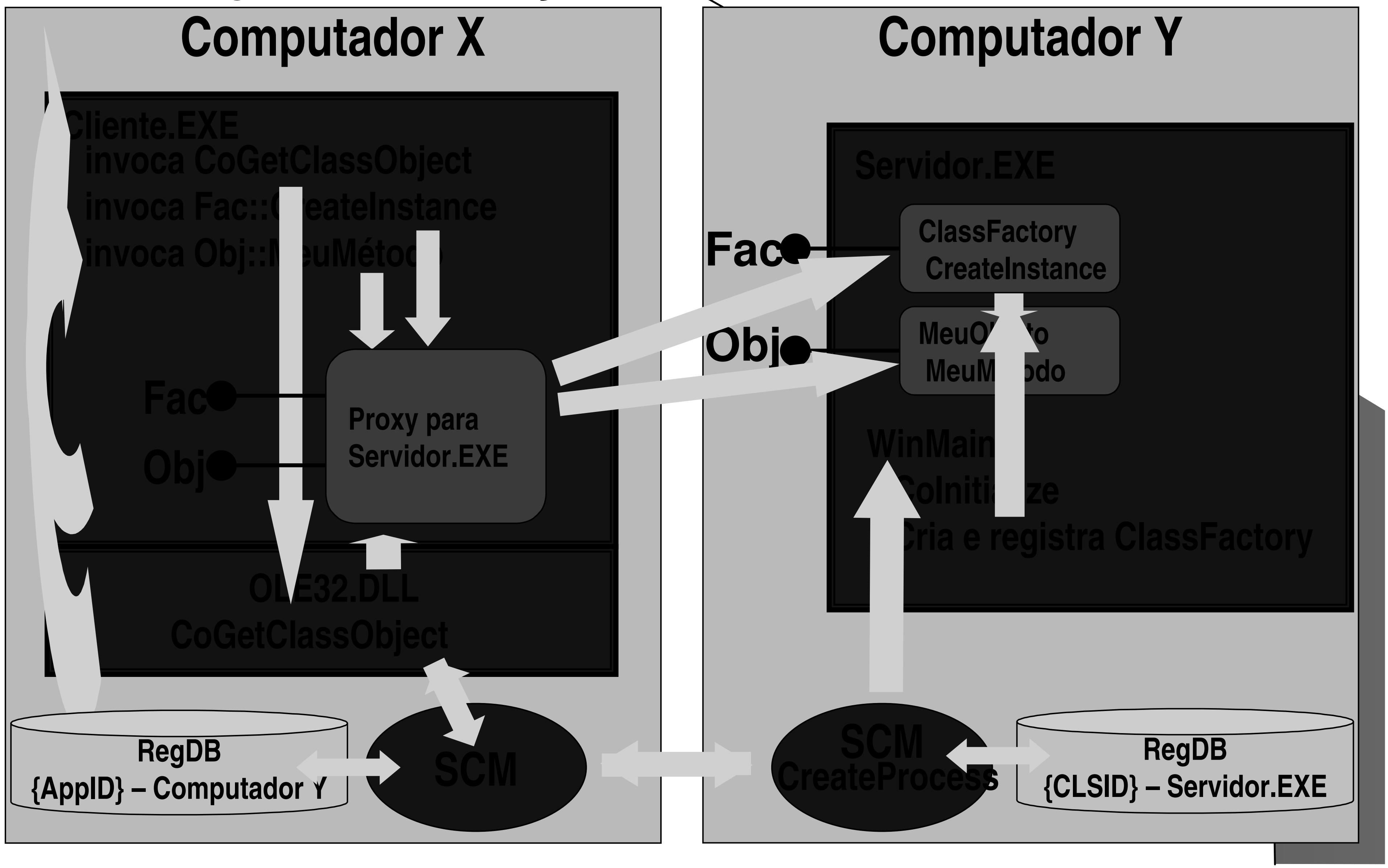


Fundamentos do COM/DCOM
Criação de Objetos – Servidor Local



Fundamentos do COM/DCOM

Criação de Objetos – Servidor Remoto



Fundamentos do COM/DCOM
Criação de Objetos - Exemplo

```
BOOL MyCheckingAccount::Init(
                const CString& name,
                const CString& serverName,
                CString& errmsg)
{
  // Create server info.
  COSERVERINFO  serverInfo;

  serverInfo.dwReserved1 = 0;
  serverInfo.dwReserved2 = 0;
  serverInfo.pwszName    =
            serverName.AllocSysString();
  serverInfo.pAuthInfo   = NULL;

  // Get the factory ...
  IClassFactory* pICF;
  HRESULT hr;

  hr = CoGetClassObject
    (CLSID_Ch3CheckingAccount, CLSCTX_LOCAL_SERVER |
     CLSCTX_REMOTE_SERVER, &ServerInfo,
     IID_IClassFactory,  (void**)&pICF);
  ::SysFreeString(serverInfo.pwszName);

  // Create the instance ...
  IUnknown* pIUk;
  hr = pICF->CreateInstance(NULL,IID_IUnknown, (void**)&pIUk);
  pICF->Release();

  // Set the interfaces with QueryInterface
  SetInterfaces(pIUk);

  // Initialize the name.
  BSTR bstrName = name.AllocSysString();
  hr = m_pIAccountInit->Init(bstrName);
  ::SysFreeString(bstrName);

  return TRUE;
}
```

Fundamentos do COM/DCOM Destruição de Objetos

- COM suporta contagem de referências e coleta de lixo distribuídos
- Quando o cliente não mais necessitar de um objeto, deve invocar o método IUnknown::Release, que decrementa o número de referências. Ao chegar a zero, o objeto é destruído
- Em ambientes distribuídos há grande chance do cliente perder comunicação com o servidor: para isso são utilizadas estratégias como *pinging/delta-pinging*, otimizadas de modo a não haver degradação significativa de performance

Fundamentos do COM/DCOM Destruição de Objetos - Exemplo

```
MyCheckingAccount::~MyCheckingAccount()
{
   Reset();
}

void MyCheckingAccount::Reset()
{
   if (m_pIAccount) m_pIAccount->Release();
   m_pIAccount = NULL;

   if (m_pIAccountInit) m_pIAccountInit->Release();
   m_pIAccountInit = NULL;

   if (m_pICheckingAccount) m_pICheckingAccount->Release();
   m_pICheckingAccount = NULL;
}
```

Referências

- Jason Pritchard. COM and CORBA Side by Side (Addison-Wesley, 1999)
- Don Box. Essential COM (Addison-Wesley, 1998)
- Dale Rogerson. Inside COM (Microsoft Press, 1997)
- David Chappell. Understanding ActiveX and OLE (Microsoft Press, 1996)
- Microsoft Developer Network, em http : // www.microsoft.com/msdn
- Alexandre Nardi. Curso “Organização e Desenvolvimento de Sistemas Distribuídos Utilizando Windows DNA” (Microsoft/Opus Software, 2000)
  - Mary Kirtland. COM+ Internals.