L’implementazione di COM

Enrico Lodolo
e.lodolo@bo.nettuno.it

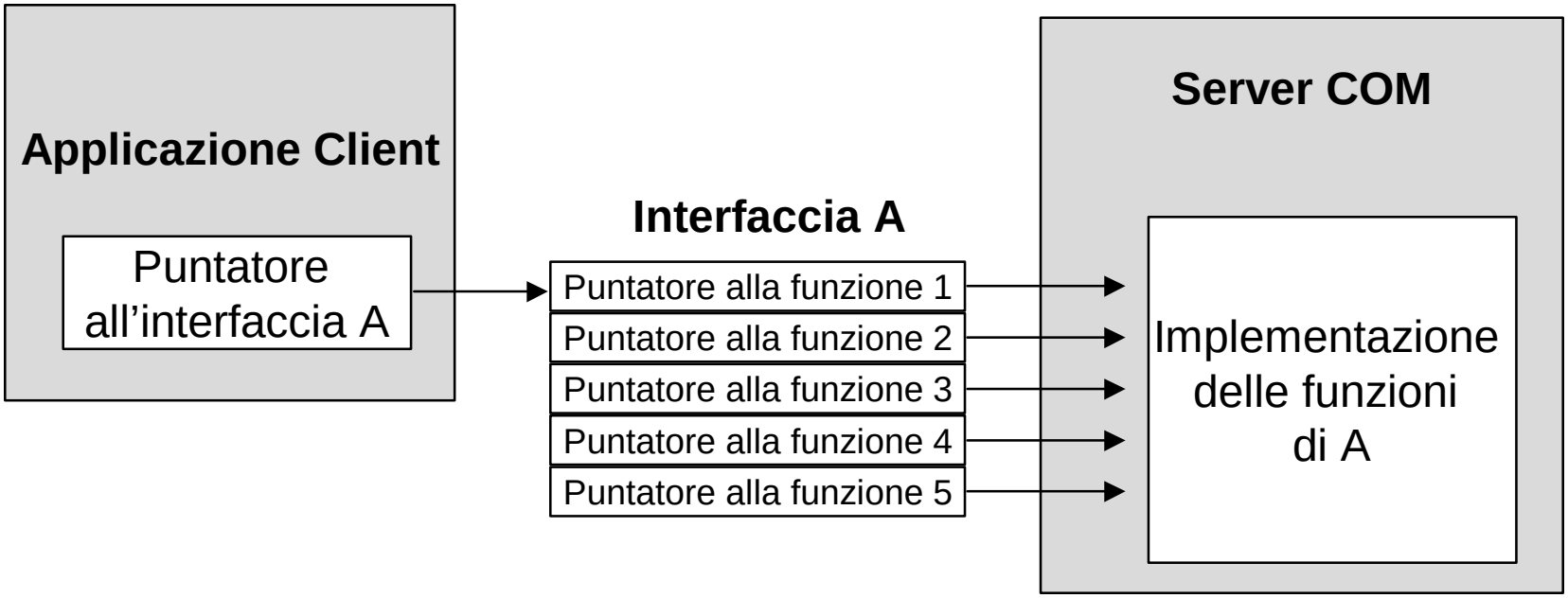
Oggetti COM e server

- **Indipendenza dalla collocazione fisica: l'applicazione che usa un oggetto COM (client) non sa dove risiede effettivamente l'oggetto**
- **Server: entità in cui risiede un oggetto COM**
- **Tre tipi di server**
  - In-process server: all'interno dello stesso spazio del processo utente, in una DLL
  - Local server: fuori dal processo ma nella stessa macchina (in un EXE)
  - Remote server: in un altra macchina: DCOM (Distributed COM)
- **I server locali e remoti vengono detti "out-of-process".**

Marshaling

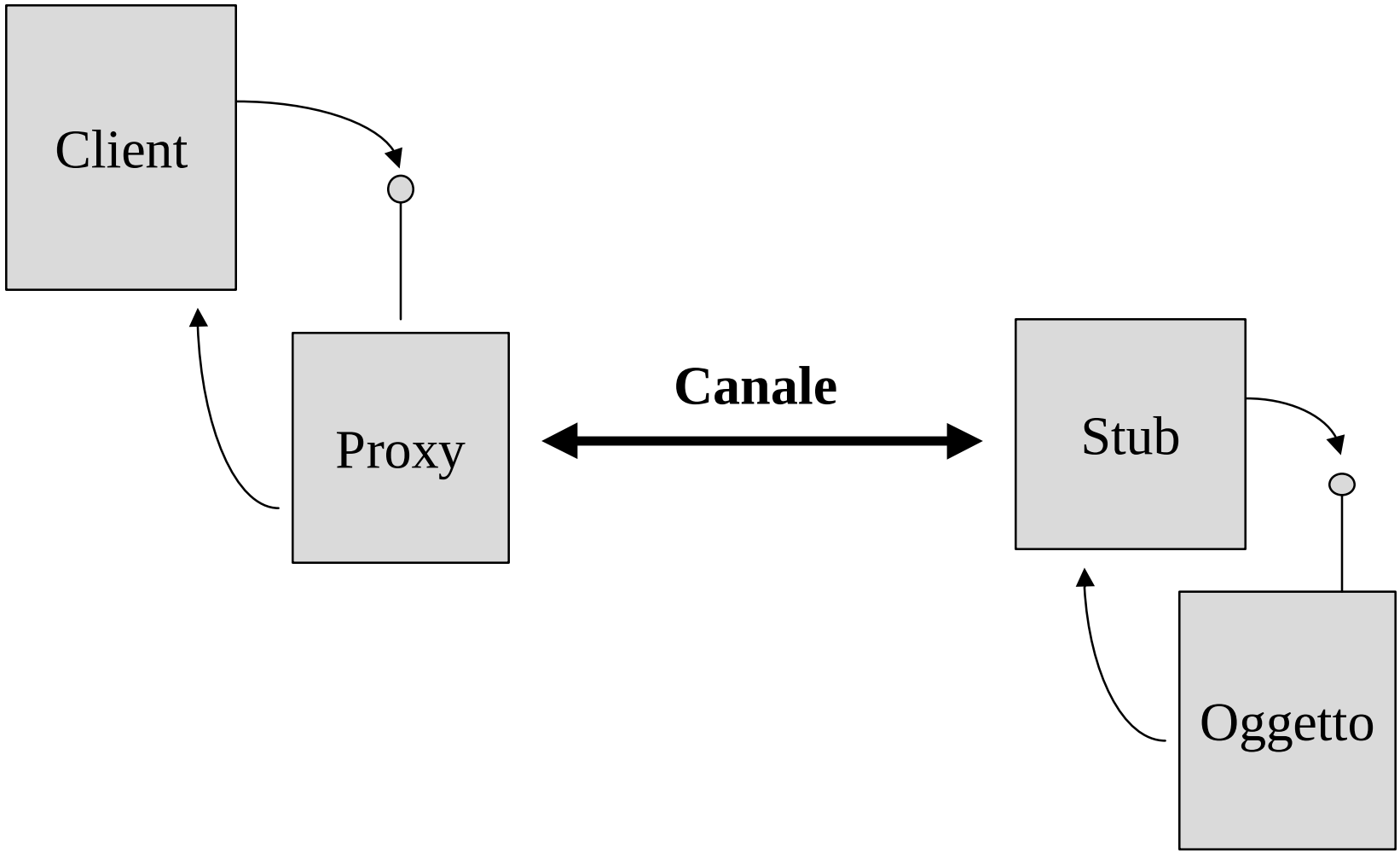
- **Marshaling: meccanismo per il passaggio delle chiamate e dei parametri fra il client e il server**
- **Il marshaling trasforma una chiamata di funzione in un pacchetto di dati (PDU: protocol data unit), contente un ID della funzione e i parametri,**
- **Questo pacchetto puòessere trasmesso con un protocollo di rete**
- **Il ricevente utilizza un meccanismo simmetrico (unmarshaling) per trasformare il PDU in una chiamata effettiva di funzione**
- **L'applicazione chiama un'immagine locale (proxy) dei metodi di un'interfaccia e il marshaling provvede a trasmettere chiamate e parametri all'oggetto effettivo**
- **In-process server: l'immagine locale coincide con l'oggetto e quindi non c'è marshaling**
- **Local server: forma semplificata di RPC, chiamata LRPC (Lightweight Remote Procedure Call) o LPC(Local Procedure Call) , basata sui messaggi Windows.**
- **Remote server: RPC standard.**

Schema del modello COM



Rappresentazione semplificata

Schema di implementazione di COM



In-process server

- **Server in-process: DLL che esporta 4 funzioni standard: DllGetClassObject, DllCanUnloadNow, DllRegisterServer, DllUnregisterServer**
- **L'applicazione client richiede un oggetto al supporto runtime (COMPOBJ) sulla base di un GUID**
- **COMPOBJ cerca nella registry il nome della DLL associata al GUID e la carica in memoria**
- **Chiama una funzione della DLL e ottiene l'interfaccia IClassFactory (vtable di un oggetto ClassFactory)**
- **Chiede a ClassFactory di instanziare l'oggetto richiesto e di restituirne l'interfaccia IUnknown (IFClassFactory.CreateInstance)**
- **L'interfaccia viene passata al client**
- **Il client invoca i metodi dell'oggetto nella DLL**

Local server: attivazione

- Il client richiede a COMPOBJ un oggetto con un GUID
- COMPOBJ ricava il nome dell'applicazione (EXE) in cui risiede l'oggetto e se necessario la avvia
- L'applicazione chiama una funzione di COMPOBJ a cui passa un puntatore alla propria interfaccia IUnknown
- COMPOBJ attiva il proxy (è un in-process server) e richiede a quest'ultimo l'interfaccia IUnknown.
- Restituisce al client l'interfaccia IUnknown del proxy
- Attiva lo stub (DLL che risiede nello spazio di memoria del server) e gli passa il puntatore all'interfaccia IUnknown del server

Local server: chiamata

- L’applicazione client chiama un metodo dell’interfaccia: in realtà è un metodo del proxy
- Il proxy esegue il “marshaling”: trasforma la chiamata e i parametri in un messaggio Windows e lo invia allo stub
- Il messaggio arriva allo stub che estrae l’identificatore della chiamata e i parametri (“demarshaling”)
- Lo stub chiama il metodo opportuno dell’interfaccia effettiva (è in pratica un callback)
- Il server esegue il metodo

Local server: ritorno

- **Il metodo termina e restituisce i risultati (valore di ritorno della funzione + parametri passati per riferimento) allo stub**
- **Lo stub trasforma i risultati in un messaggio Windows (marshaling) e lo invia al proxy**
- **Il proxy riceve il messaggio ed estrae i risultati**
- **Il metodo del proxy termina e restituisce i risultati al client**

Remote server

- **Il meccanismo è lo stesso del local server**
- **Il passaggio dei messaggi avviene però su una rete attraverso un meccanismo RPC standard (DCE)**
- **Il marshaling consiste nella creazione di un PDU (Protocol Data Unit) ovvero un pacchetto di dati da trasmettere sulla rete, per esempio come pacchetto TCP/IP**
- **Nel proxy e nello stub abbiamo cicli di attesa che rendono sincrona l'operazione**
- **L'aspetto più complesso è garantire che, pur in presenza di un meccanismo sincrono, le applicazioni non si blocchino (deadlock)**

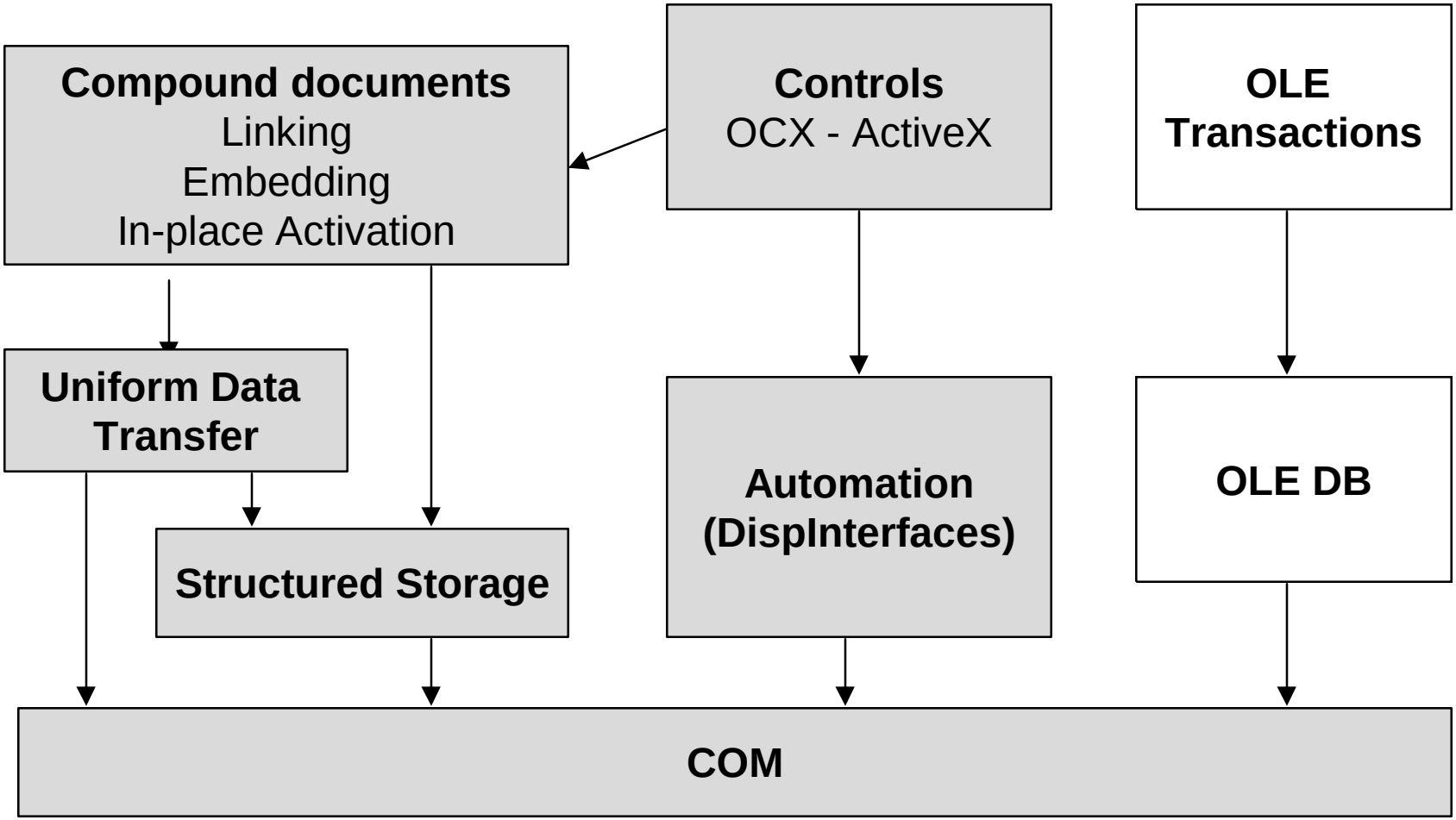
Generalizzazione del marshaling

- Ogni interfaccia o insieme di interfacce richiede una coppia proxy/stub in grado di eseguire marshaling/demarshaling
- In una prima fase COM metteva a disposizione questo servizio solo per le interfacce standard
- La Microsoft sconsigliava la definizione di interfacce non standard
- L'uso di un'interfaccia non standard richiedeva la scrittura manuale di un proxy e di uno stub dedicati
- Recentemente è stato introdotto un meccanismo generalizzato

MIDL

- **Le interfacce vengono descritte mediante un apposito linguaggio (MIDL)**
- **Il compilatore MIDL genera automaticamente la coppia proxy/stub a partire da una di queste descrizioni**
- **MIDL è un linguaggio con una sintassi simile a quella del C ed è compatibile con lo standard DCE: IDL (Interface Definition Language)**
- **MIDL = Microsoft Interface Definition Language**
- **Incorpora un meccanismo precedente (ODL=Object Definition Language) che forniva una soluzione parziale legata ad OLE Automation**

Le tecnologie ActiveX



Documenti composti (OLE)

- **Integrazione di componenti provenienti da fonti diverse all'interno di un documento composto**
- **Container: applicazione che gestisce il documenti**
- **Server: applicazioni specializzate che gestiscono i componenti**
- **Embedding: il componente viene incorporato nel documento**
- **Linking: il documento contiene solo un riferimento al componente**
- **In-place activation: l'applicazione server si attiva nello spazio visivo (finestra) del container**

Structured storage

- Consente di creare un intero file system all'interno di un singolo file
- Gestione di concorrenza e diritti di accesso
- Gestione delle transazioni
- Nato come supporto per il salvataggio dei documenti composti
- Base per il futuro file system ad oggetti (OFS) di Cairo

Uniform data transfer

- **Unifica i vari sistemi di scambio dati fra processi: clipboard, drag and drop, DDE**
- **Indipendenza dal mezzo fisico usato per lo scambio (memoria, file, protocolli di rete ...)**
- **Separazione netta fra impostazione del trasferimento (protocollo) ed effettivo scambio dati**
- **Lo scambio avviene mediante un particolare oggetto COM denominato Data Object**
- **Il protocollo è costituito da una serie di funzioni API che hanno il compito di scambiare un Data Object fra due processi**

Automation

- **Consente ad un'applicazione (controller o client) di comandare dall'esterno un'altra applicazione (server)**
- **E' in pratica un metodo di scripting generalizzato indipendente dal linguaggio.**
- **E' la tecnologia più usata con DCOM (Remote automation)**
- **Consente di incapsulare e riutilizzare applicazioni legacy**
- **Fornisce un metodo semplificato per creare interfacce non standard senza ricorrere a proxy/stub**