Software Development with Components

The Microsoft Component Object Model (COM)

Egil P.Andersen
Norwegian Computing Center
P.O.Box 114, Blindern, 0314 Oslo, Norway
Tel: +47 22 85 25 94, Fax: +47 22 69 76 60
Egil.Paulin.Andersen@nr.no

Slides:	http://www.nr.no/~egil/hit-component-101000.ppt
Source code:	http://www.nr.no/~egil/hit-sourcecode-101000.zip
Run demo:	http://www.nr.no/~egil/hit-rundemo-101000.zip

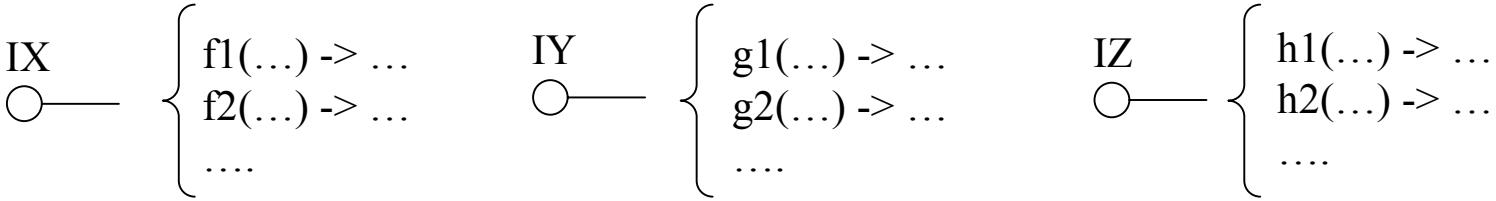
Programming Languages and Development Environment

- **Microsoft Visual Studio** - an elaborate development environment
- **Visual Basic** - very(!) easy to learn and use - inflexible - performance
- **Visual C++** - powerful and flexible - complex - wizzardmania….
- **Visual J++** - no experience with it…..
- **ATL (Active Template Library)** - utility for creating COM components in VC++

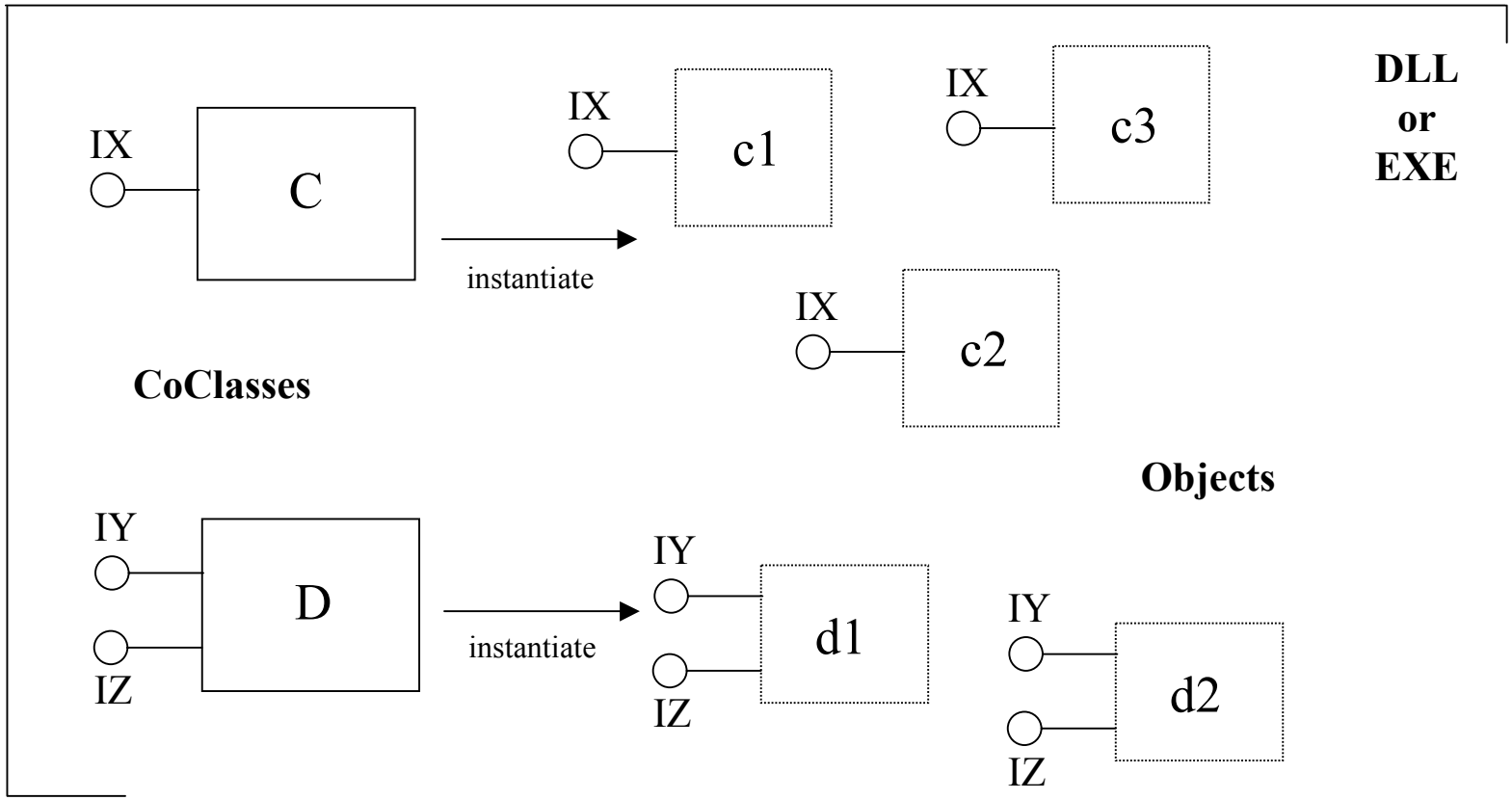
Key Issues for Software Components

- **Naming** - GUID's (Globally Unique Identifiers), Prog.ID, Type Libraries, Registry
- **Life-Cycle Management** - Reference counting
- **Programming Language Independence** - COM is a binary standard. See memory layout below
- **Location Transparency** - In-process, local out-of-process, remote out-of-process. Distributed COM (DCOM). Not 100% - preferably the same organisation in charge of both client and server.
- **Versioning** - Published interfaces are (should be…) immutable.
- **Extensibility** - Multiple interfaces per component. Component aggregation.

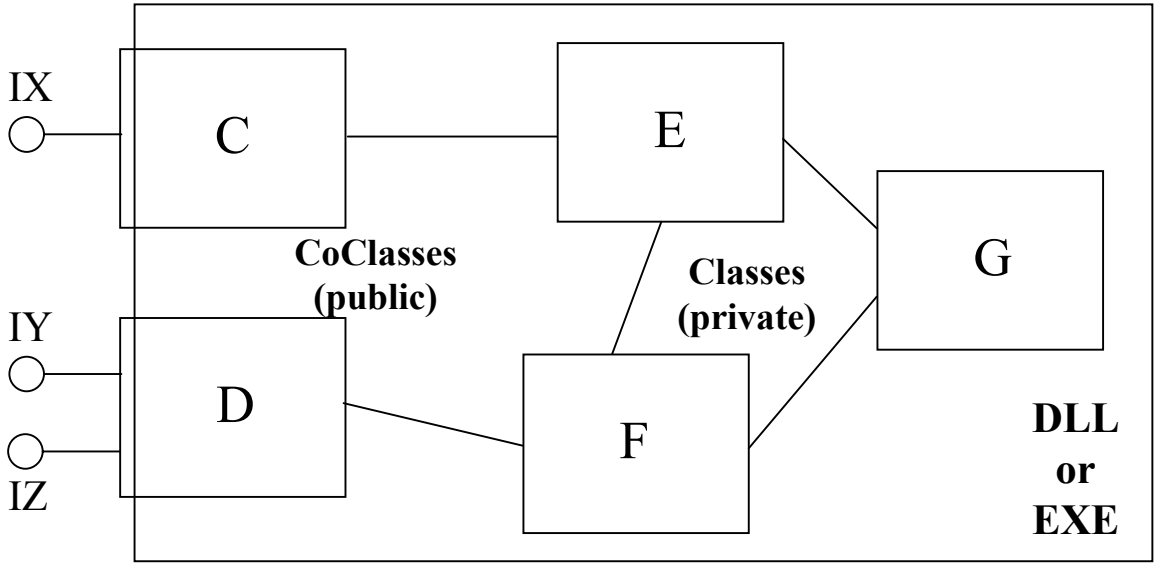
Interfaces, Components/CoClasses, Objects, GUID (Globally Unique Identifiers), CLSID, IID



Interfaces: Versioning - Multiple interfaces - Single inheritance - IUnknown



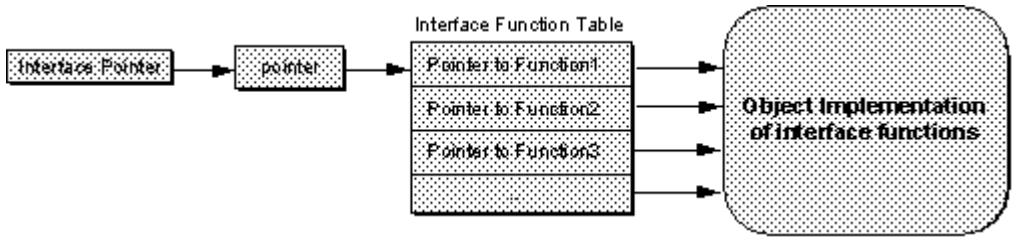
Public CoClasses vs Private Classes
Instantiable CoClasses



IDL - Interface Definition Language

```
[ object,
  uuid(EA762187-A99A-11d3-95F4-0060979B4844),
   oleautomation,
   dual,
   …..]
interface IOSSSMLogin : IDispatch
{[id(1), helpstring("Function LogOn")]
    HRESULT LogOn([in] BSTR user, [in] BSTR pwd, [out] VARIANT_BOOL* okLogOn);

 [id(2), helpstring("Function LogOff")]
    HRESULT LogOff([out] VARIANT_BOOL* okLogOff);
};

[ object,
   uuid(EA762188-A99A-11d3-95F4-0060979B4844),
   oleautomation,
   dual,
   ….]
interface IOSSSMXML : IDispatch
{ [id(1), helpstring("Function GetRecordInfo")]
    HRESULT GetRecordInfo([in] long recordID, [in] short retrievalMode,
                                  [in] VARIANT_BOOL getHTML, [out] BSTR* XMLString);
    ….. };
```

Basic COM (Component Object Model)



- **VTable interfaces** - a binary standard with interfaces based on a memory layout corresponding to that of abstract classes in C++

  A COM interface and its functions is similar to an abstract base class with a set of virtual functions in C++

  The extra level of indirection provides flexibility with respect to how interfaces are implemented.

- **Dispatch interfaces** - query the interface for its functions and their signatures

- **Dual interfaces** - available both for efficient vtable access and for scripting languages

Information on Interfaces and Components

OLE/COM Object Viewer

File Object View Help

- Objekt på egenskapsside for in
- ObjTest Class
- Office Compatible 1.0
- Office Graphics Filters Thumbn
- OfficeWebServer SyncMgr Ha
- Offline Pages Cleaner
- OLE DB Rowset Proxy
- OLE DB Rowset Server
- OLEViewer Automation Interfac
- OLEViewer Default Interface V
- OLEViewer IPersist Interface V
- OLEViewer IUnknown Interfac
- OLEViewer TypeLib Interface V
- OLEViewer UDT Interface View
- Option Propery Page 2.1
- OptionObj Object
- OptionsCollection Object
- OPWNoLink
- OrderListExport
- ORG10SVR 1.0 Application
- ORG21SVR 1.0 Application
- OSSCOMInterface.IServerLog(
- OSSCOMInterface.IServerXML
- OSSCOMInterface.ISMLogOn
- OSSCOMInterface.ISMXML
- Outlook CalendarView
- Outlook CardView
- Outlook Central
- Outlook DocSite OLE Control

Ready

No icon Available

OSSCOMInterface.ISMLogOn
{C38A7CEF-A50C-11D3-8C1A-0431F7C00000}

Registry | Implementation | Activation

CLSID = {0000031A-0000-0000-C000-000000000046}
{C38A7CEF-A50C-11D3-8C1A-0431F7C00000} = OSSCOMInterface.ISMLogOn
ProgID = OSSCOMInterface.ISMLogOn

ITypeLib Viewer

File View

- OSSCOMInterface
  - dispinterface _ISMLogOn
  - interface _ISMLogOn
    - LogOn
    - LogOff
    - Inherited Interfaces
      - IDispatch
        - GetTypeInfoCount
        - GetTypeInfo
        - GetIDsOfNames
        - Invoke
        - Inherited Interfaces
          - IUnknown
            - QueryInterface
            - AddRef
            - Release

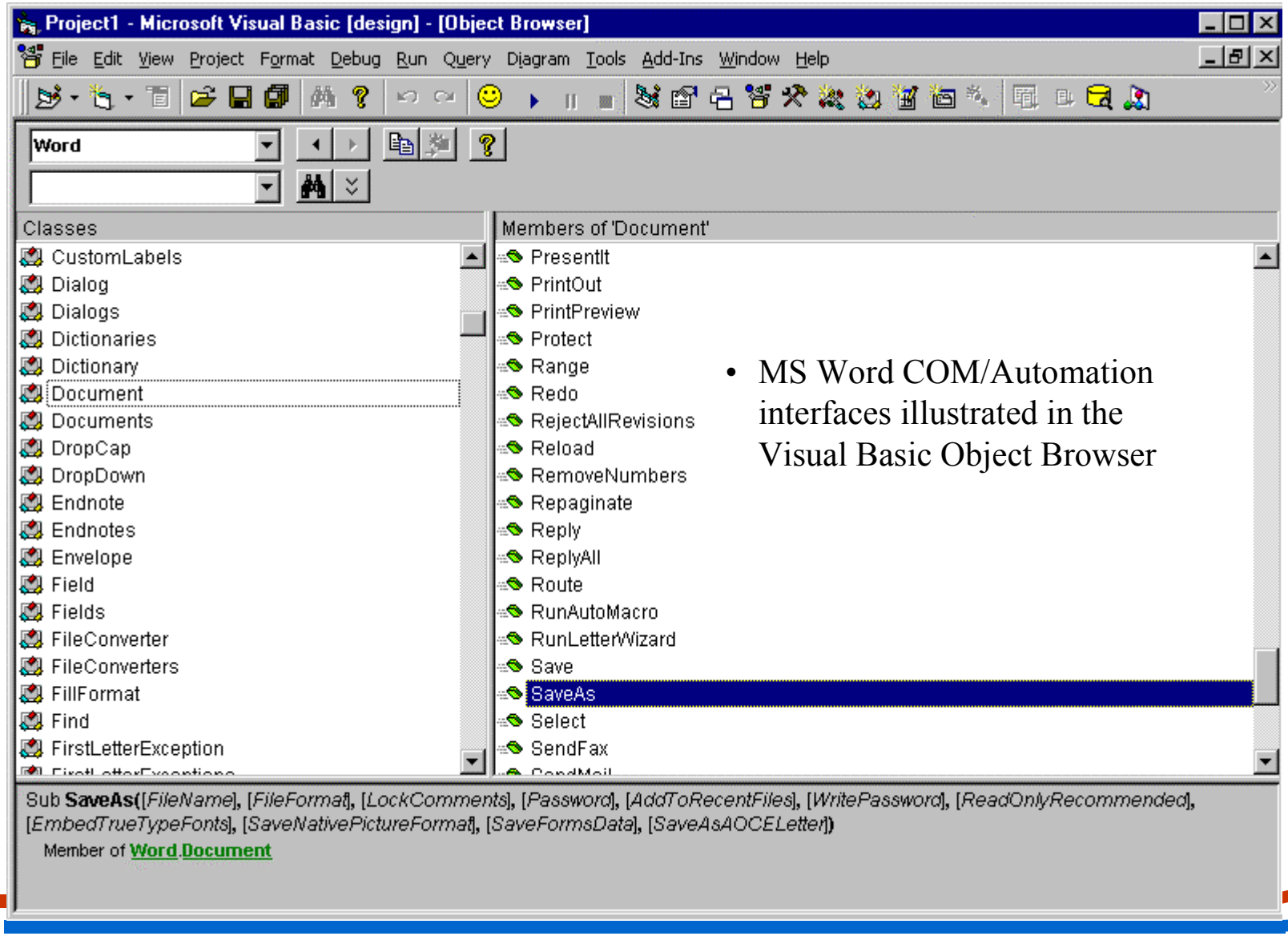
```
// Generated .IDL file (by the
OLE/COM Object Viewer)
//
// typelib filename: <could not
determine filename>

[
  uuid(5CF2244D-86C3-11D3-95DD-
0060979B4844),
  version(1.0)
]
library OSSCOMInterface
{
    // TLib :     // TLib : OLE
Automation : {00020430-0000-0000-
C000-000000000046}
    importlib("StdOle2.Tlb");

    // Forward declare all types
defined in this typelib
    interface _ISMLogOn;
```

Ready

Start | Utforsker - C:\... | CD-ROM-spiller | Volume Control | Microsoft Visu... | Microsoft Pow... | OLE/COM ... | 22:16

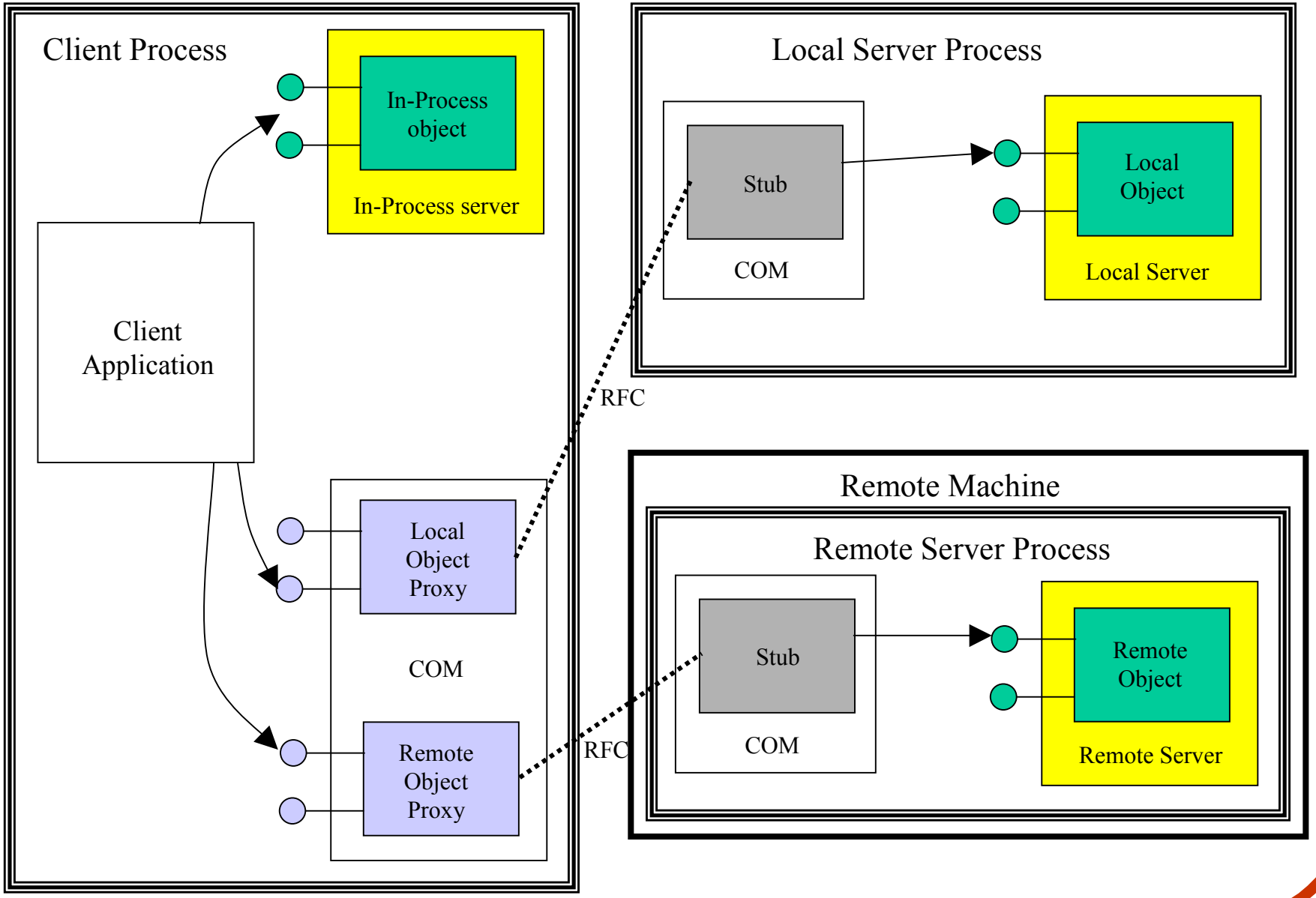
Component Object Models

- In component based systems an object model consists of classes, interfaces, functions, etc, typically specified by an IDL (interface definition language).

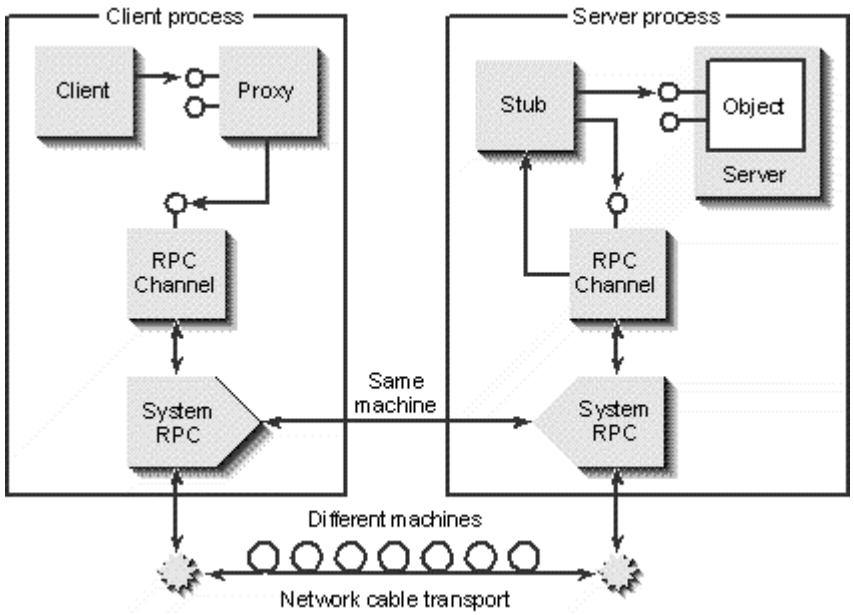


- MS Word COM/Automation interfaces illustrated in the Visual Basic Object Browser

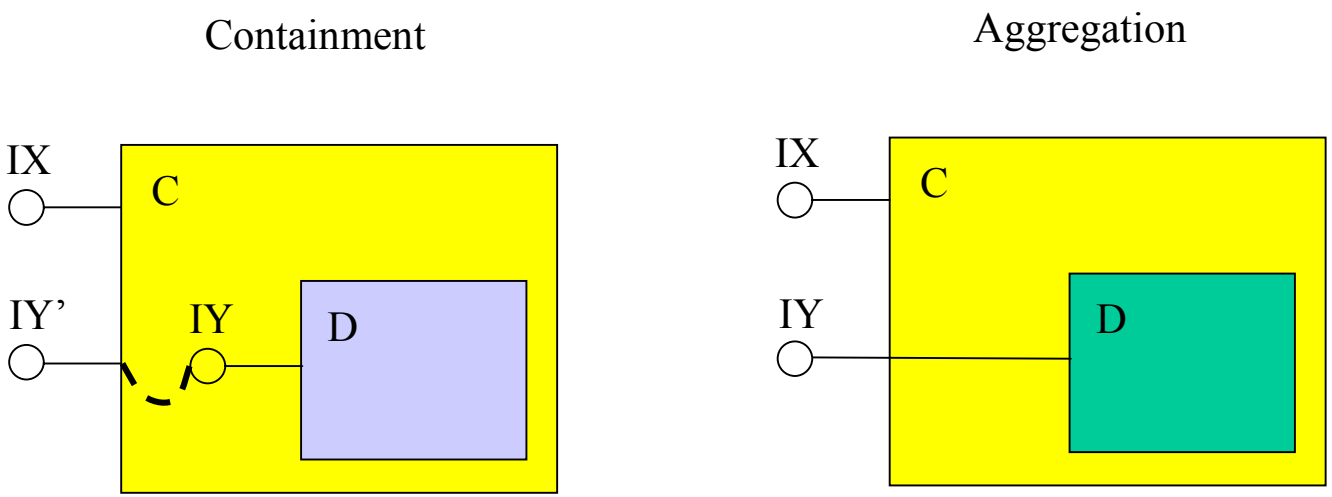
Local in-process, Local out-of-process, Remote



Marshalling for Out-of-Process Components



Integrating COM Components via Containment vs Aggregation



Compound Documents

with ActiveX Controls and ActiveX Documents

Testing Word spellchecking

Check spelling | splell | Incorrect spelling

Goto | http:\\www.usit.uio.no

Universitetet i Oslo Søk Oppslag Hjelp

Universitetet i Oslo

USIT - Universitetets senter for informasjonsteknologi

USIT informerer

- IT-tjenester ved UiO
- USITs elektroniske oppslagstavle
- Avisa Info@usit.uio.no

Project1 - Microsoft Visual Basic [run] - [Form1 (Code)]

File Edit View Project Format Debug Run Query Diagram Tools
Add-Ins Window Help

cmdCheck | Click | Procedure

```
Option Explicit

Private Sub cmdCheck_Click()
    Dim myWord As Word.Application
    Set myWord = New Word.Application
    If myWord.CheckSpelling(txtSpell.Text) Then
        lblResult.Caption = "Correct spelling"
    Else
        lblResult.Caption = "Incorrect spelling"
    End If
End Sub

Private Sub cmdGoto_Click()
    WebBrowser1.Navigate (txtURL.Text)
End Sub
```

Example - “4 in a row”

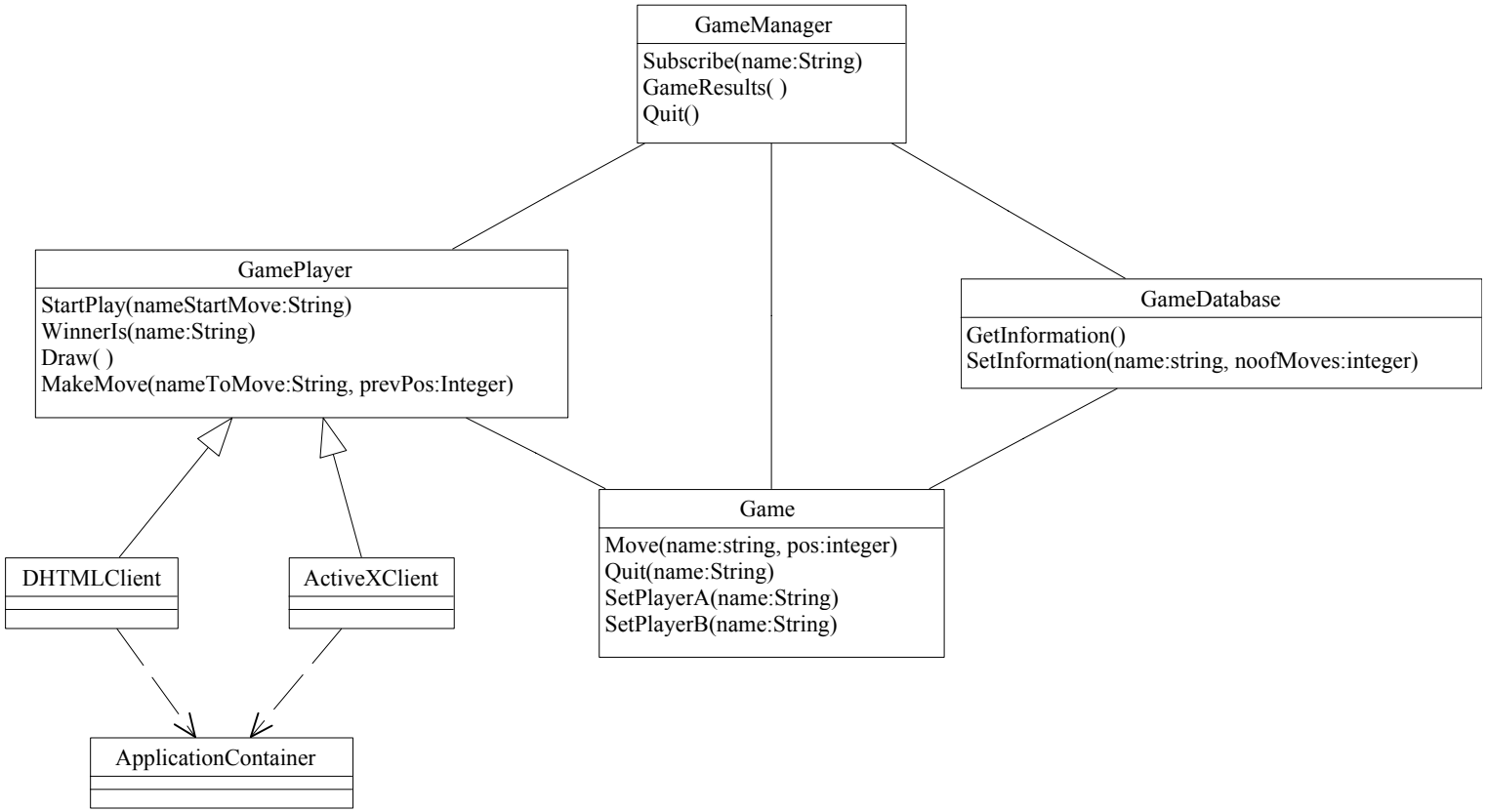
Simple multi-player game

(but the current demo version is not made distributed. With *SOAP (Simple Object Access Protocol)* is can very easily be made to run over a web server…)

Components

- **GameDatabase** - Handles game results. Visual Basic .dll. Uses ADO and XML.
  Single interface specified by separate IDL file.
- **GameManager** - Manages clients that want to play a game. Visual Basic .exe (because will be shared between several clients).
  Multiple interfaces specified by “virtual” Visual Basic classes.
- **Game** - Component shared by clients playing together. Visual C++ .dll (with ATL).
  Multiple interfaces, both ingoing and outgoing (connection points/events), in IDL.
- **ActiveX Client** - ActiveX control for game presentation and interaction. Visual Basic .ocx.
- **DHTML Client** - HTML page for game presentation and interaction that utilises Dynamic HTML (DHTML) and VBScript (or JavaScript/ECMAscript)
- **Application Container** - Container for running a game either with the ActiveX Client or with the DHTML Client within a web browser control.

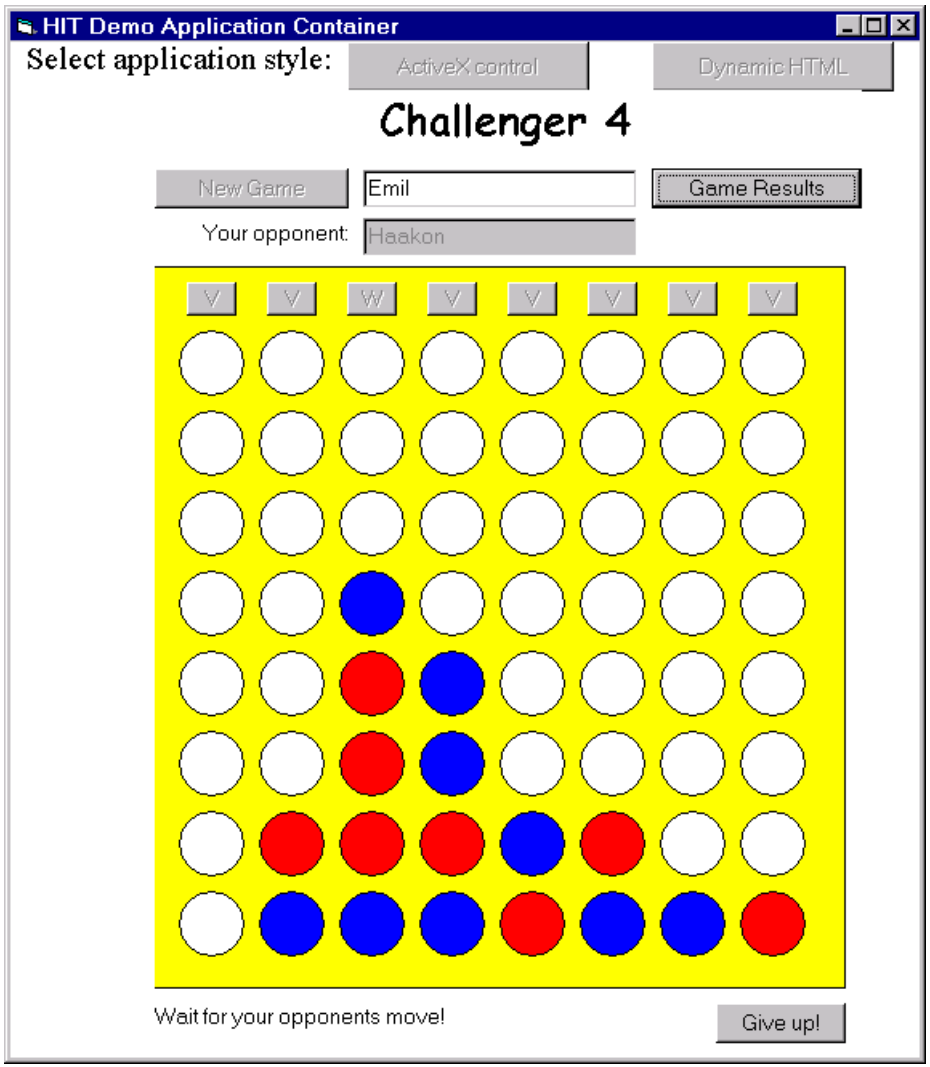
Example - “4 in a row” (cont.)



PS: This example is made for demonstration purposes only - to demonstrate how to implement components and how to specify their interfaces. The design itself has many flaws - for example, outgoing interfaces/events should in general not be used between server-side and client-side components.

Example - “4 in a row” (cont.)

Client as ActiveX control



Client as DHTML script