Component Technology and Distributed Information Systems on the Internet

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

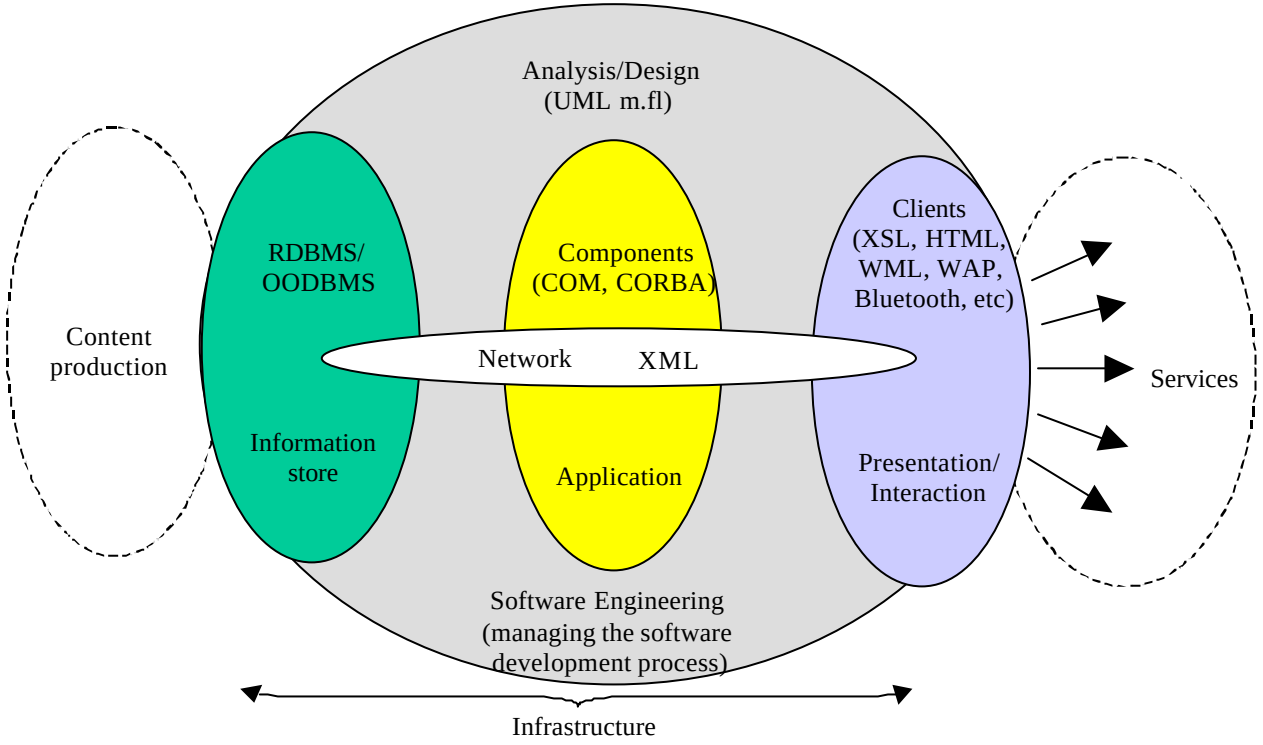
Some Tools and Technologies....

Some tools and technologies for the development of Component based Information Systems

- Object Management Architecture, CORBA (Common Object Request Broker Architecture)
- Microsoft COM (Component Object Model), DCOM (Distributed COM)
- Component Object Models, IDL (Interface Description Language)
- Layered Architectures, 3-tier/n-tier, Business Objects
- Rational Rose/UML (Unified Modelling Language)
- Visual Basic, Visual C++, ATL (Active Template Library), J++ (MS Java)
- Compound Documents, ActiveX Controls, ActiveX Documents
- Universal Data Access, OLE DB, ADO (ActiveX Data Objects)
- MTS (Microsoft Transaction Server)
- IIS (Internet Information Server), ASP (Active Server Pages), Scripting
- SOAP - Simple Object Access Protocol
- XML, DTD (Document Type Definition), DOM (Document Object Model), XSL (eXstensible Stylesheet Language)
- Microsoft Repository, Visual Component Manager (VCM)
- Microsoft Message Queue Server (MSMQ)
- OODB (??)
- …..and more….puh….

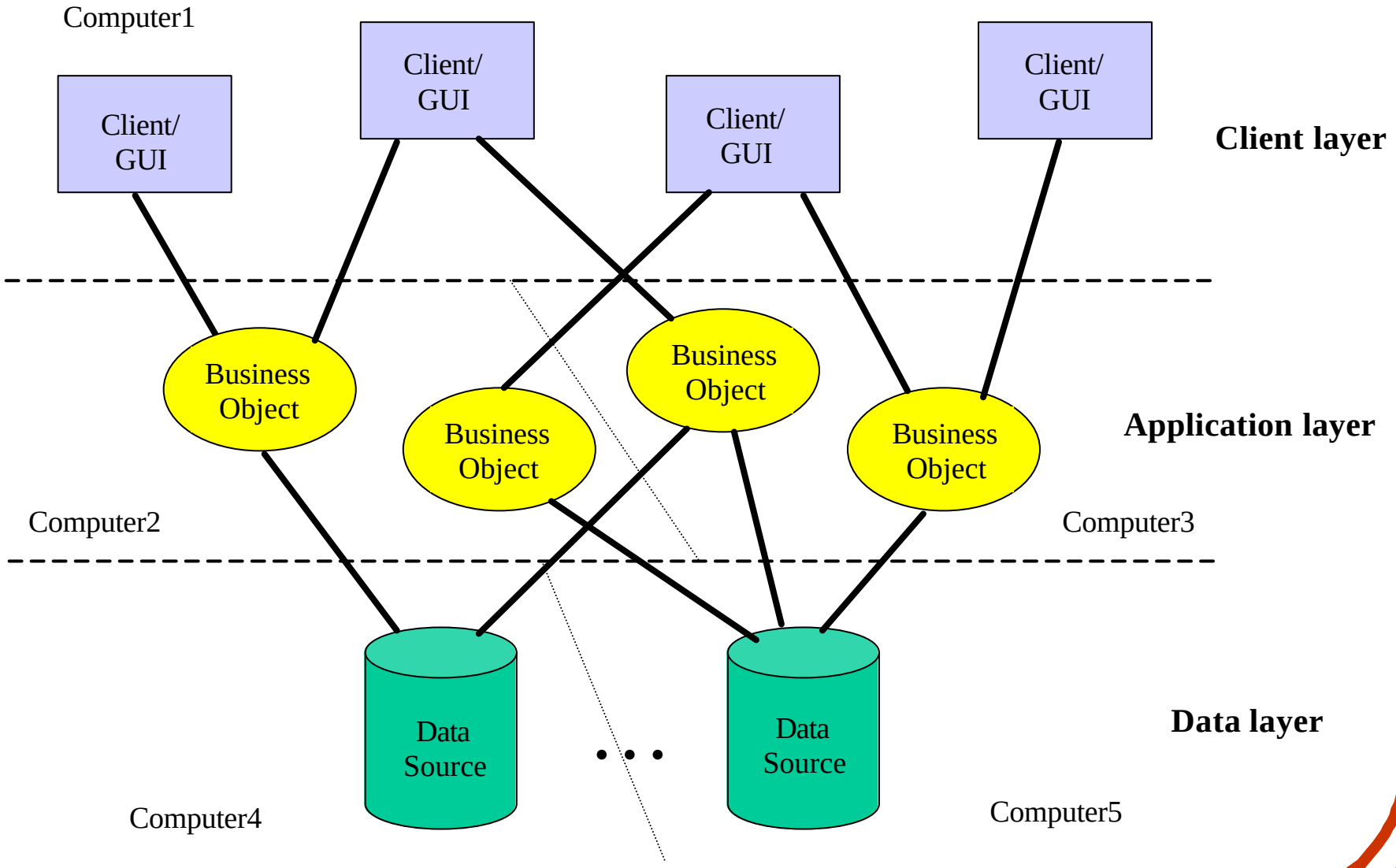
Distributed Information Systems

Information Systems - An Outline



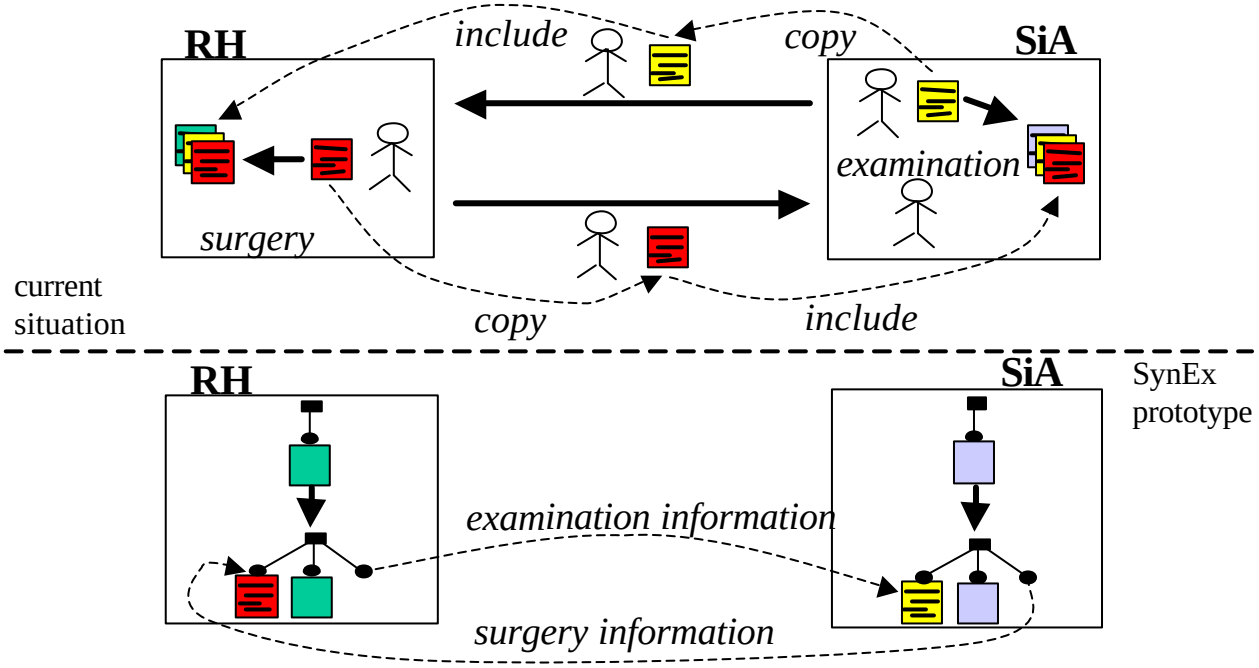
Layered Architectures - 3-tier/n-tier

Windows DNA (Distributed interNet application Architecture)



SynEx - Synergy on the Extranet

Seamless Integration of Distributed Electronic Patient Records

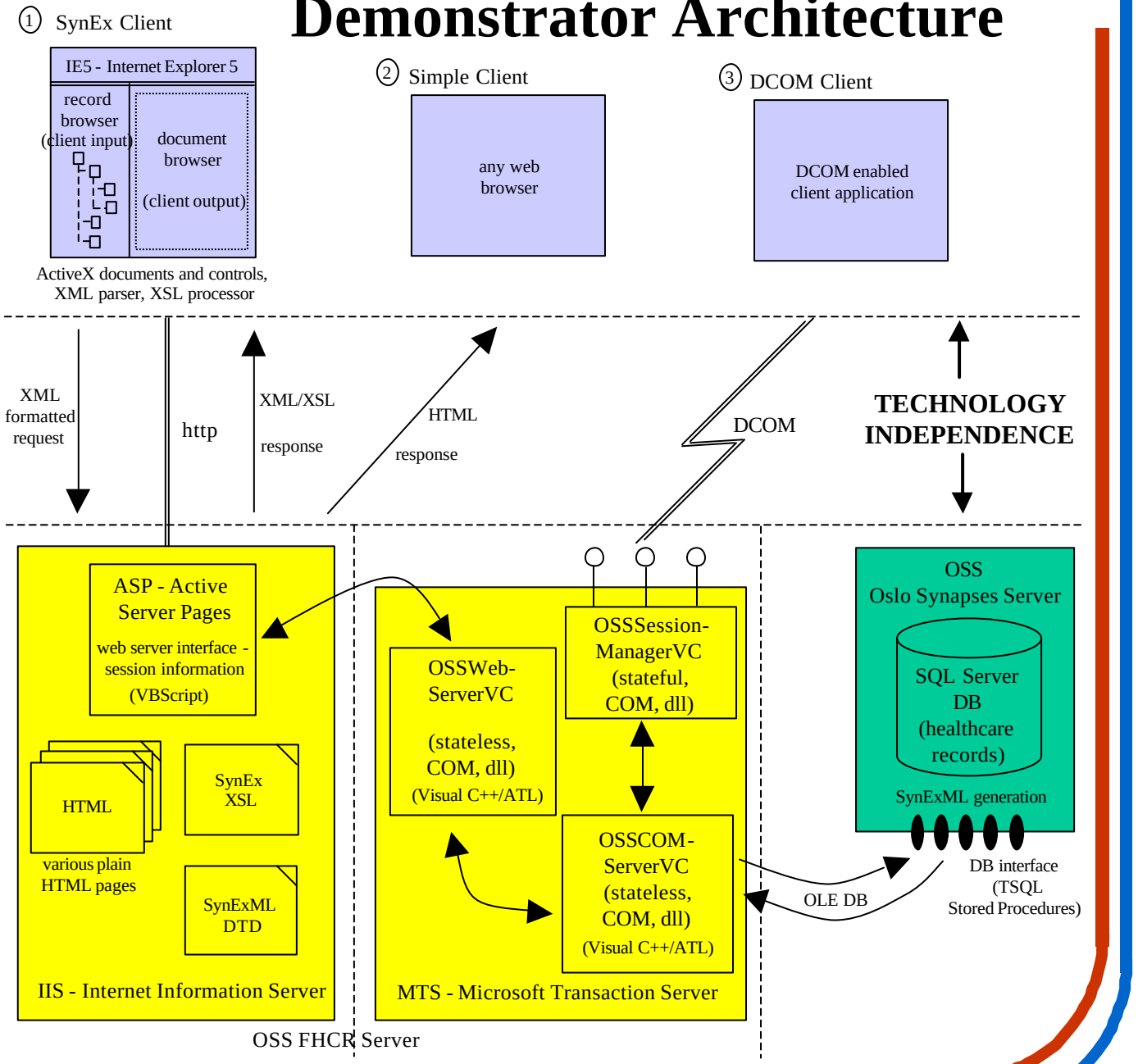


Demonstrator Architecture

SynEx Client:
ActiveX components in Internet Explorer

Request/Response:
XML over http

OSS FHCR Server:
IIS/ASP on Win/NT and COM comp. under MTS



Information Storage

Databases - Persistent Information

- Databases and database technology play a key role in information management systems

  Techniques and methods for efficient (w.r.t. searching) storage of large volumes of persistent information made available to (many) concurrent users, and
  operations on this information requires the administration of distributed transactions.

- Professional web sites will use databases for storing their information, while the individual web pages are generated "on demand".
  This as opposed to an unmanageable "mix" of .html, .xml and .wml files.

  Except for reading/browsing, then the information content of .html/.xml/.wml files is not readily available for comparisons, search, etc.
  In addition there are the problems of invalid and missing links.

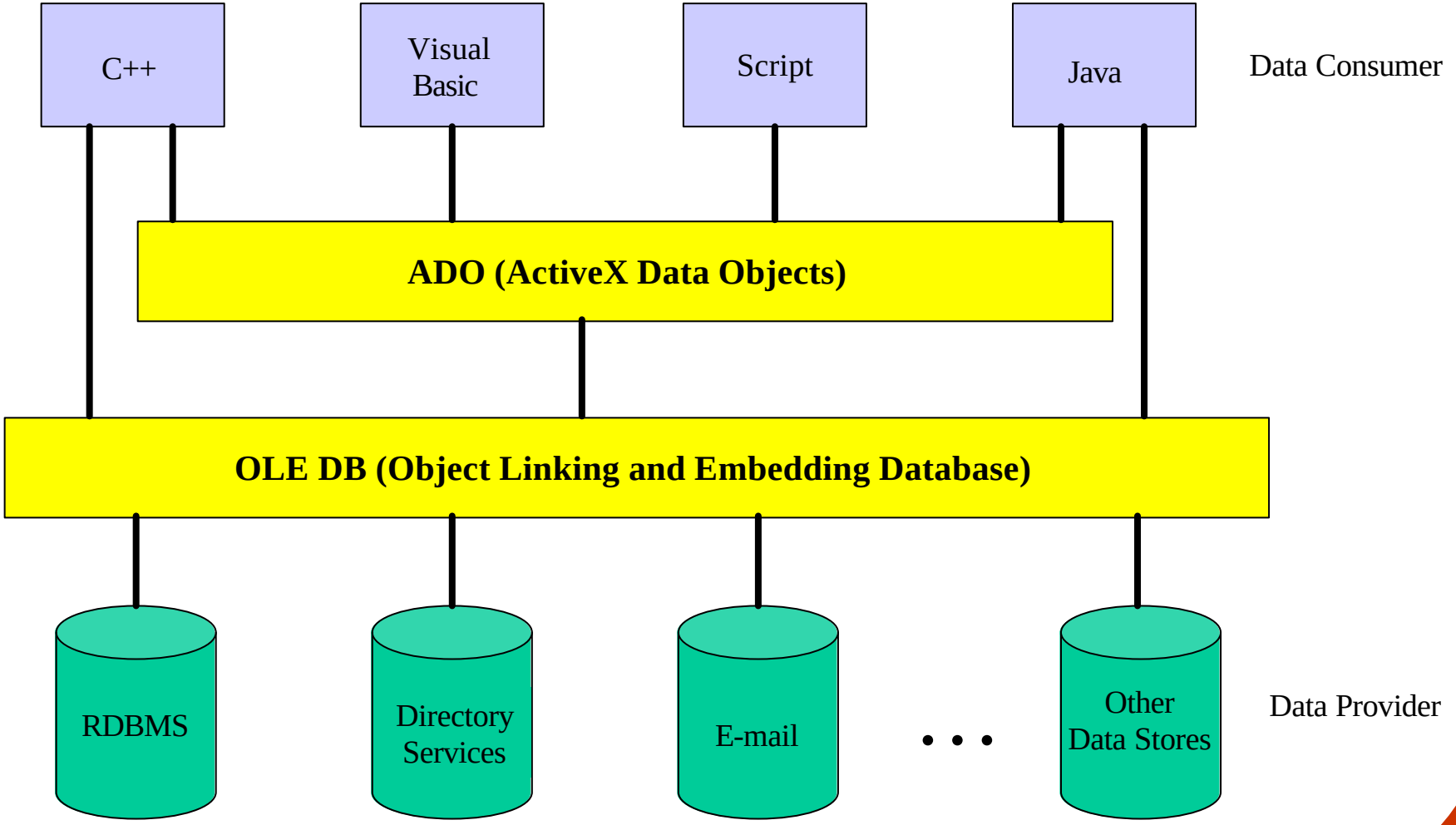
Stored Procedures and Database Encapsulation

Encapsulate the database tables behind an "interface" of stored procedures

- Constraints and business rules that are inherently linked to the database schema, independent of which application uses the database, should be enforced within the database, and stored procedures may be the only means for achieving this.
- The tables of a database schema are often subject to minor changes, e.g. for performance reasons, but such changes should be transparent to the application.
- Improved performance

Universal Data Access (UDA)

Uniform Access to Heterogenous Data Sources with OLE DB and ADO



What about Object-Oriented Databases?

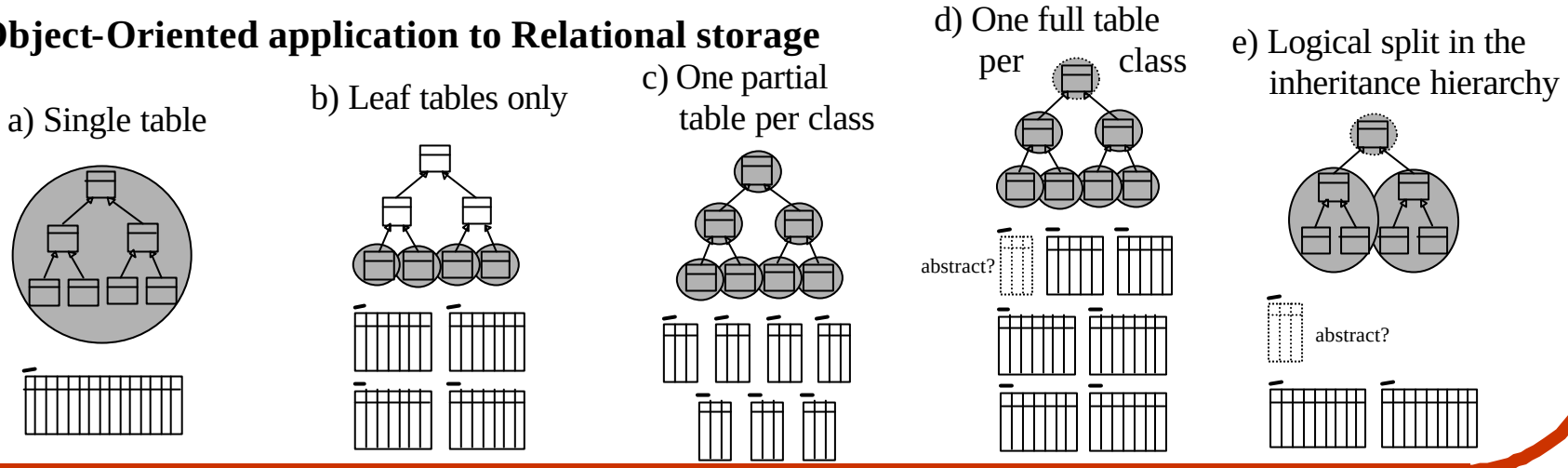
- Main benefit

  Avoids mismatch between relational data and object-oriented applications;
  e.g. inheritance relationships, recursive structures, ….

- Do they scale well, do they perform well - may be - but many in industry considers it an added risk to rely on this for large enterprise information systems
- Developers will be happy with them - but will this reduce development cost enough to outweight the “risk” (real or perceived)?
- What about customers - do they benefit from it?
- Main problem

  No “killer application” - there seems to be no undisputable need for it

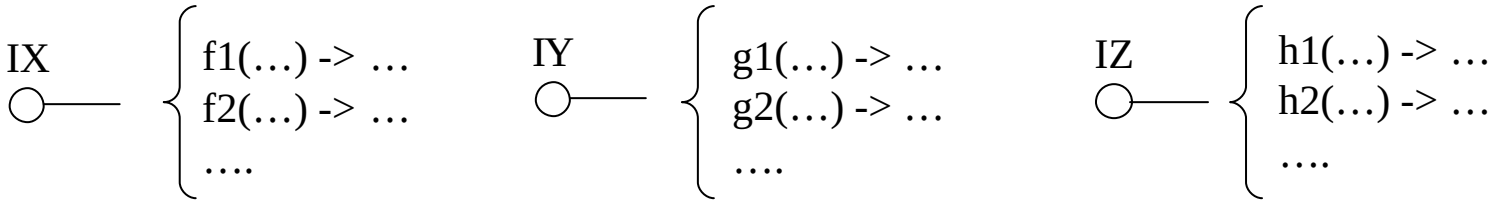
Object-Oriented application to Relational storage



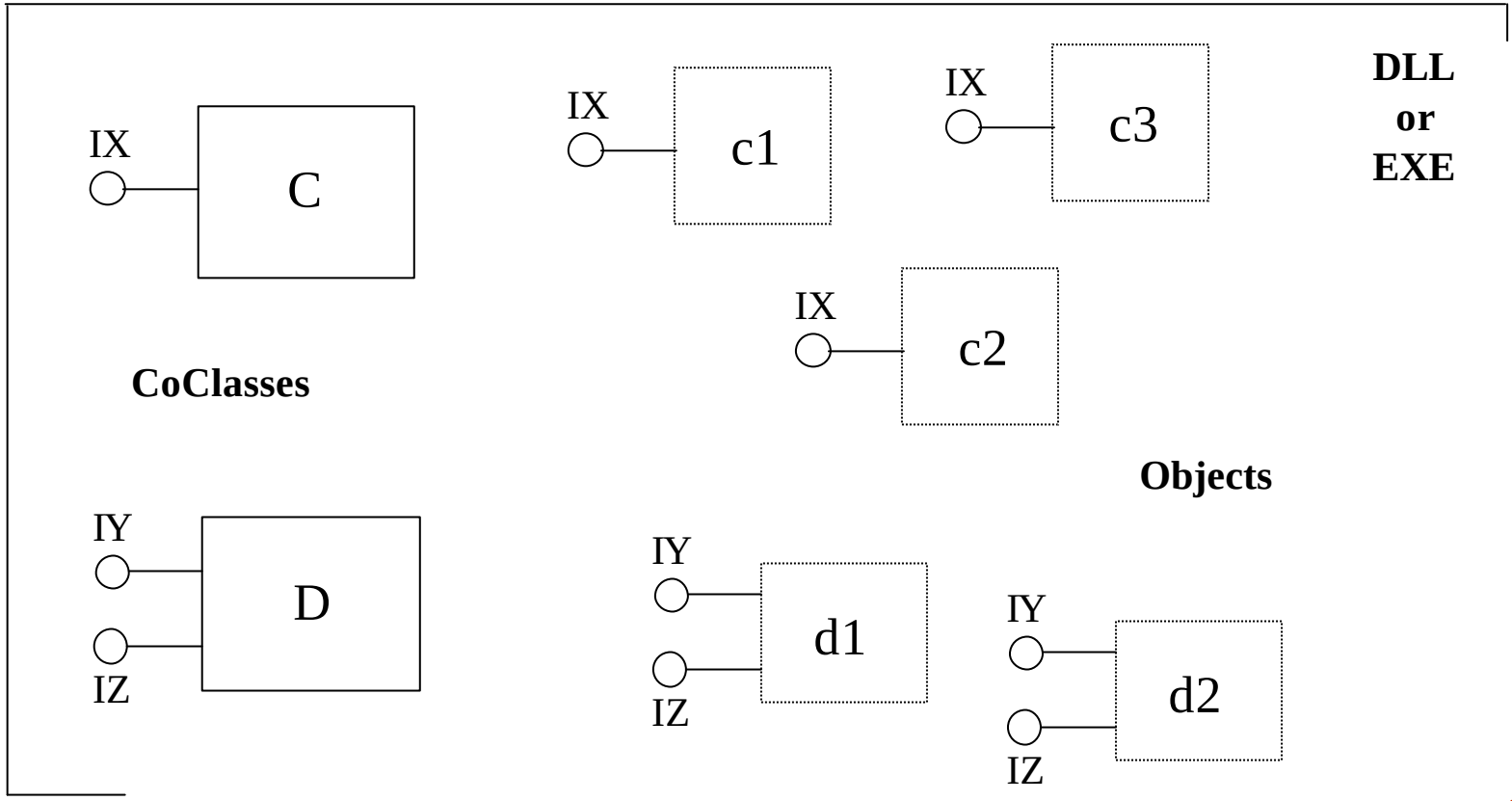
COM Components

(Microsoft Component Object Model)

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



Interfaces: Versioning - Multiple interfaces - Single inheritance - IUnknown



IDL - Interface Definition Language

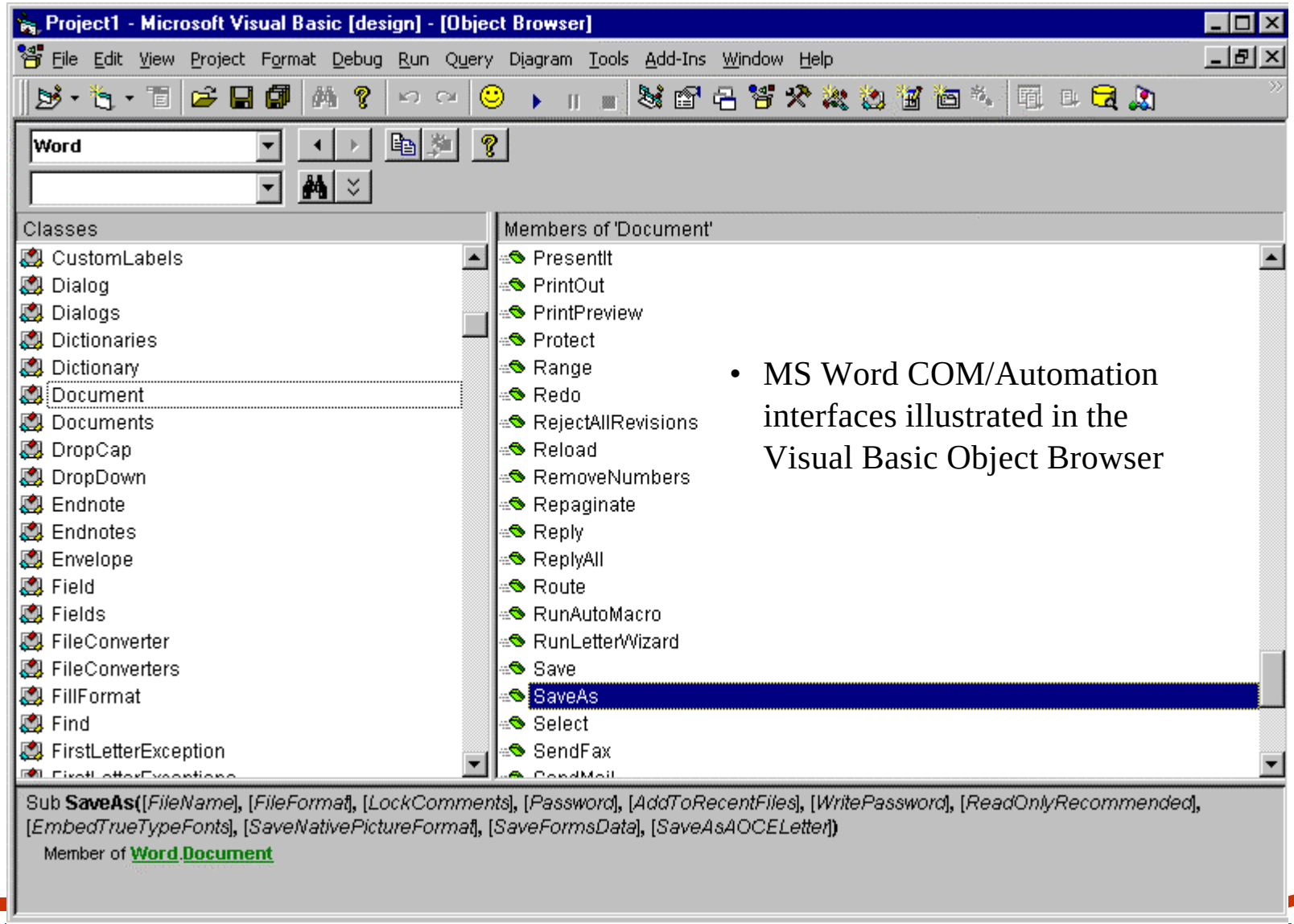
```
[ object,
  uuid(EA762187-A99A-11d3-95F4-0060979B4844),
   oleautomation,
   dual,
   …..]
interface IOSSSMLogin : IDispatch
{[id(1), helpstring("Function LogOn")]
    HRESULT LogOn([in] BSTR user, [in] BSTR pwd, [out] VARIANT_BOOL* okLogOn);

 [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);
   ….. };
```

Component Object Models

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



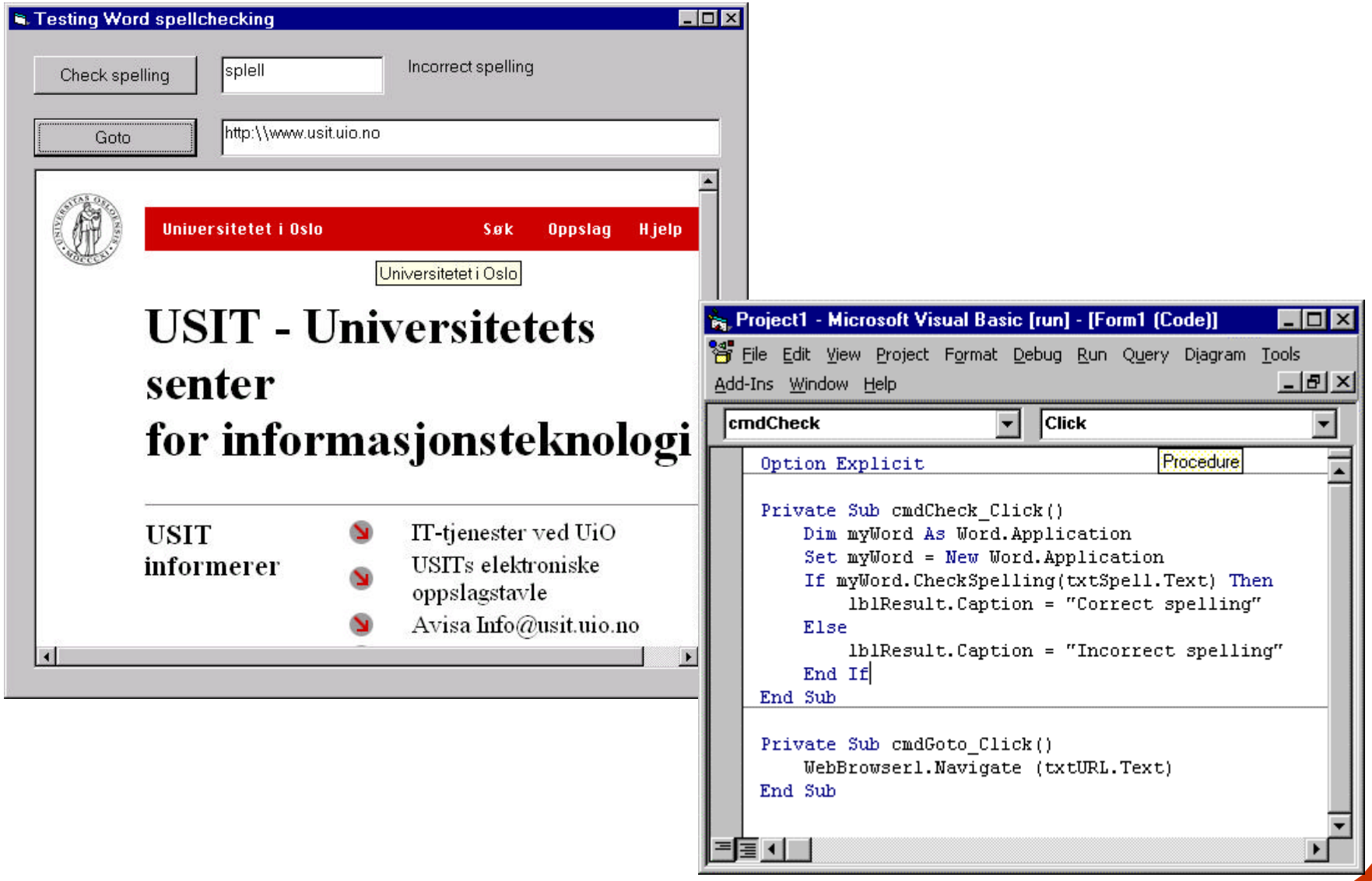
- MS Word COM/Automation interfaces illustrated in the Visual Basic Object Browser

Programming Languages and Development Environment

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

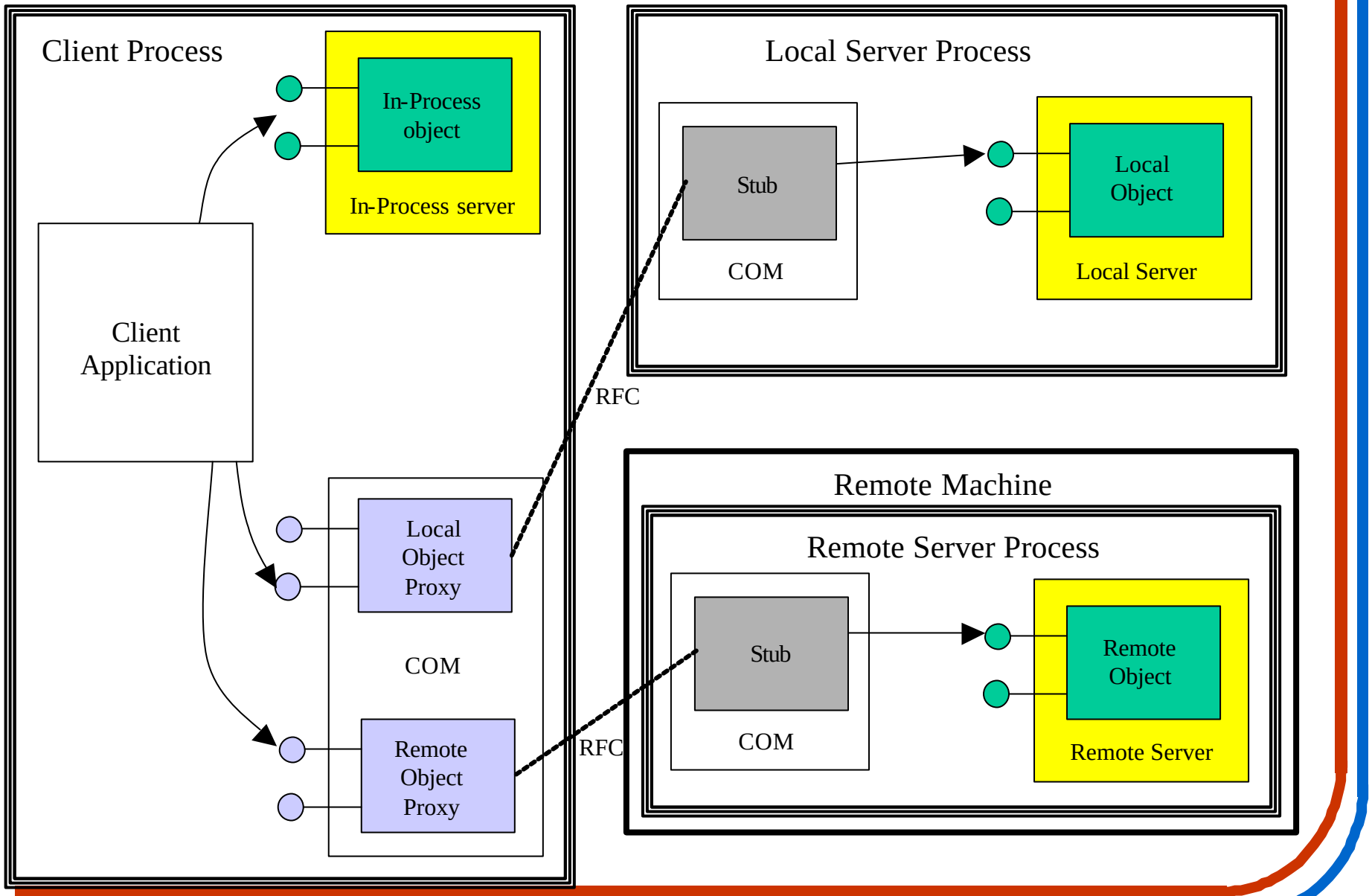
Compound Documents

with ActiveX Controls and ActiveX Documents



COM and Distributed COM (DCOM)

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



Application Layer

MTS - Microsoft Transaction Server

Transaction servers are important for *scalability* with respect to the number of concurrent users, and thus *performance*, and also for managing *distributed transactions* and *resources* like database connections. MTS - the transaction server from Microsoft - supports:

- Distributed transactions via DTC (Distributed Transaction Coordinator)

  COM objects residing in the MTS of different computers can participate in the same atomic transaction.
  If an MTS COM object works on databases on different computers these database operations can be combined into a single transaction.

- Database connection pooling

  Instead of assigning a dedicated database connection to each client, a pool of database connections are reused as required to serve client requests => database connections can be utilized more efficiently

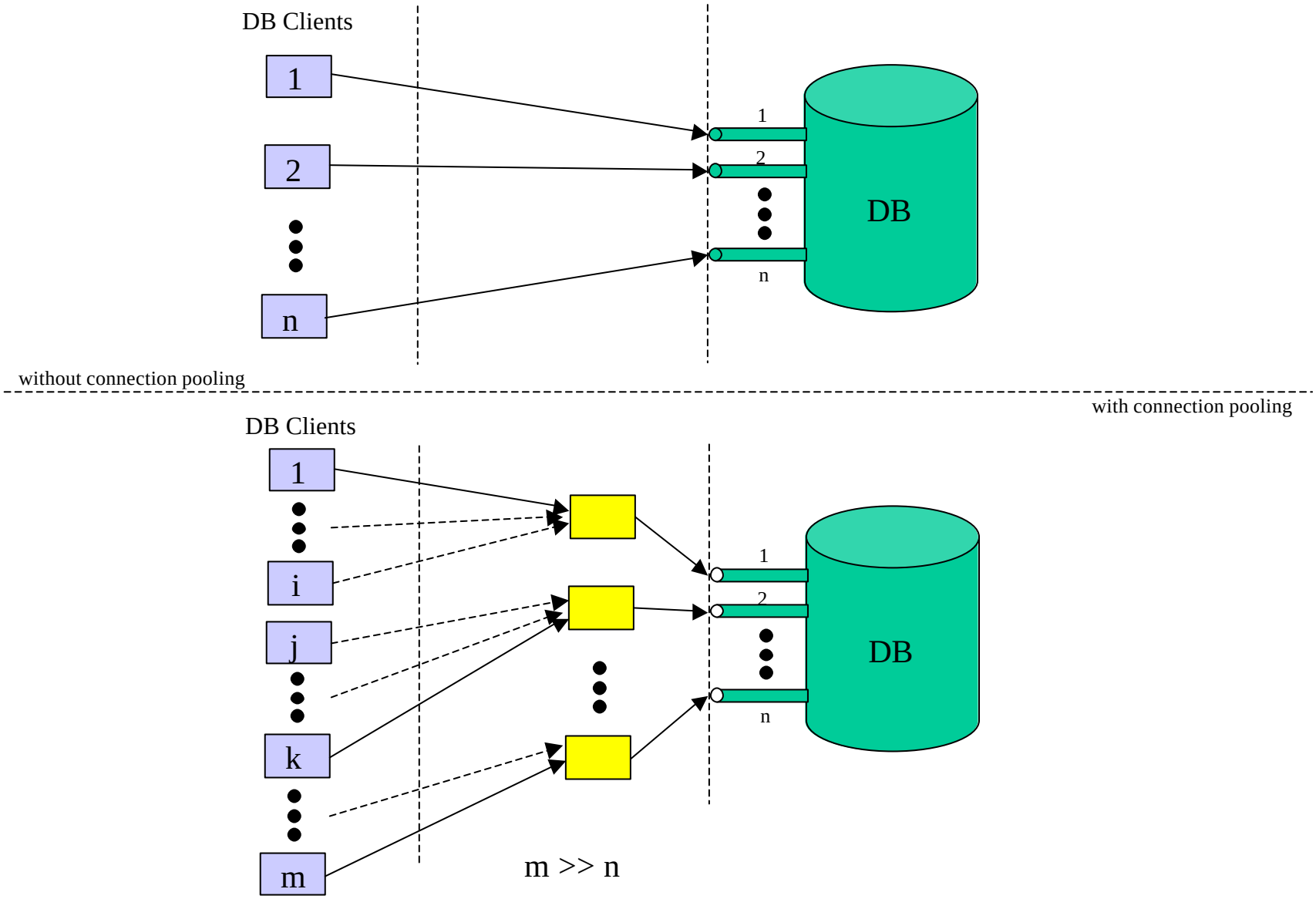
- Object pooling (only available from MTS v.3.0) - "Stateless" programming model

  Instead of creating an object instantiated from a particular component from scratch each time it is needed, objects that does not participate in a transaction at the moment can be reused as if they were newly created objects.

  "Stateless" COM objects - they may well have state within transactions but not between transactions

COM+ - the latest version of COM - MTS included
ATL (Active Template Library) - VC++ library that makes it easier to implement COM components

MTS Connection and Object Pooling



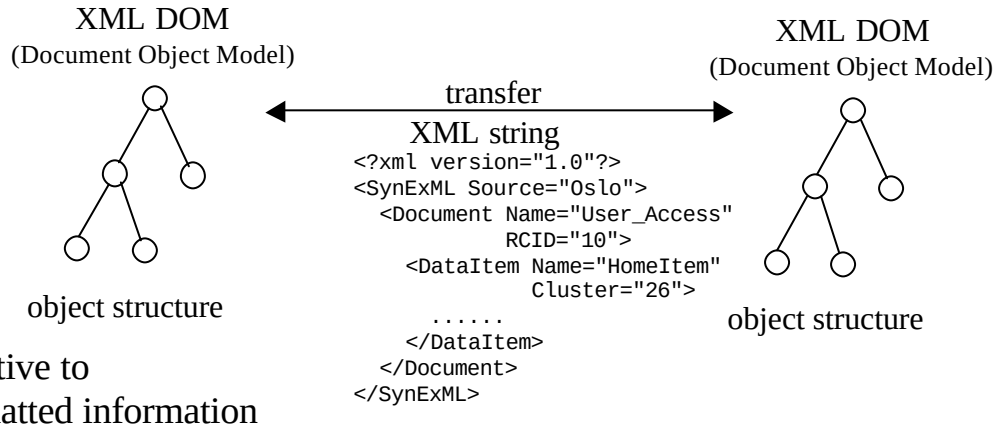
XML

(eXtensible Markup Language)

XML - eXtensible Markup Language

What is XML?

- XML is a string of text formatted according to certain rules.
  Some of the format rules are common to all XML (*well formed XML*),
  while others can be defined by an XML schema definition (*valid XML)* - e.g. an *XML DTD (Document Type Definition), XML Schema*, and others.
- An XML string can be stored in a plain ASCII file, but when using XML in an information system the XML may never exist in a file.
- Important - When creating or receiving an XML string it can be accessed and operated upon as a structure of various kinds of objects with an interface with functions (and also events) similar to other kinds of e.g. COM objects.



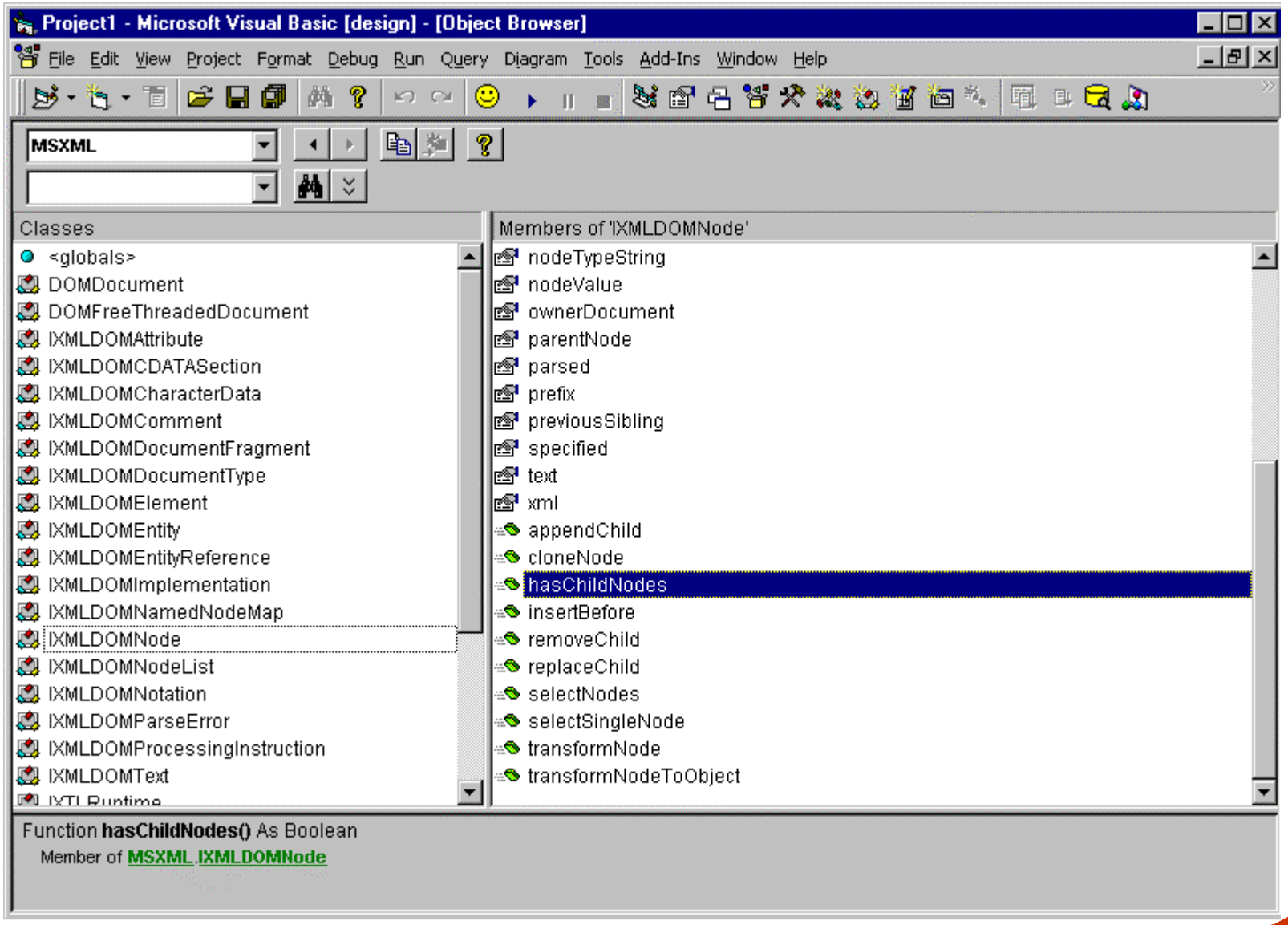
What is XML **not** (at least primarily...)?

- A modelling language
- For storing persistent information
  i.e., not principally different from how any file can be used for this. Thus no alternative to databases for large data volumes. XML formatted information is well-suited for database storage - generated on demand - as opposed to more spesialised data formats.
- For presentation
  XML in combination with XSL (eXstensible Stylesheet Language) can be used for flexible presentation formats - e.g. into HTML or other presentation formats.

WAP/WML on Mobile Devices

- WML (Wireless Markup Language) is XML according to a particular DTD

Document Object Model (DOM) of the MS XML Parser



SOAP - Simple Object Access Protocol

Microsoft currently works on a specification called *SOAP (Simple Object Access Protocol)* where the communication between a client and a server is formatted as XML over http both ways.

There are several advantages by this:

- http is a simple protocol with good coverage and few demands on the client

  XML as strings are well-suited for transmission via http

- Most firewalls are readily configured for common security options dealing with well known internet protocols and ports.

  This as opposed to e.g. DCOM or CORBA protocols like IIOP (Internet Inter-ORB Protocol).

  In practice, the ability for remote machines to interact via DCOM and IIOP is more limited.

  DCOM and IIOP can be well-suited for computers within e.g. a limited area, but not between "any" remote client and server on the internet.

- XML over http makes the underlying client- and server-side technology transparent to each other.

  Similar to how component technology provides for programming language independence and technical interoperability locally, SOAP provides for platform independence and technical interoperability globally.

Example SOAP Requests

http GET command (QueryString)

```
http://citroen.nr.no/synexdemo/oss.asp?<OSSrequest>
                                        <Function Name="LogOn">
                                        <Arg Name="User">admin</Arg>
                                        <Arg Name="Password">x</Arg>
                                        <Arg Name="ResponseType">xml</Arg>
                                        </Function>
                                      </OSSrequest>
```

http POST command (HTML Forms)

```
<FORM METHOD="POST" ACTION="http://citroen.nr.no/synexdemo/oss.asp">
      <INPUT TYPE="hidden" NAME="XMLRequest"
                           VALUE='<OSSrequest><Function Name="LogOn">
                                    <Arg Name="User">emil</Arg>
                                     <Arg Name="Password">x</Arg>
                                   <Arg Name="ResponseType">xml</Arg>
                                    </Function></OSSrequest>'/>
      <INPUT TYPE="submit" VALUE="Log On"/>
</FORM>
```

Notice: The above XML format is *not* according to the SOAP v.0.9 specification.

Web Server
IIS/ASP
(Internet Information Server/Active Server Pages)

ASP scripts as Web Server Interfaces

- Avoid the use of scripting languages (e.g. VBscript) except as ”glue” between COM components.

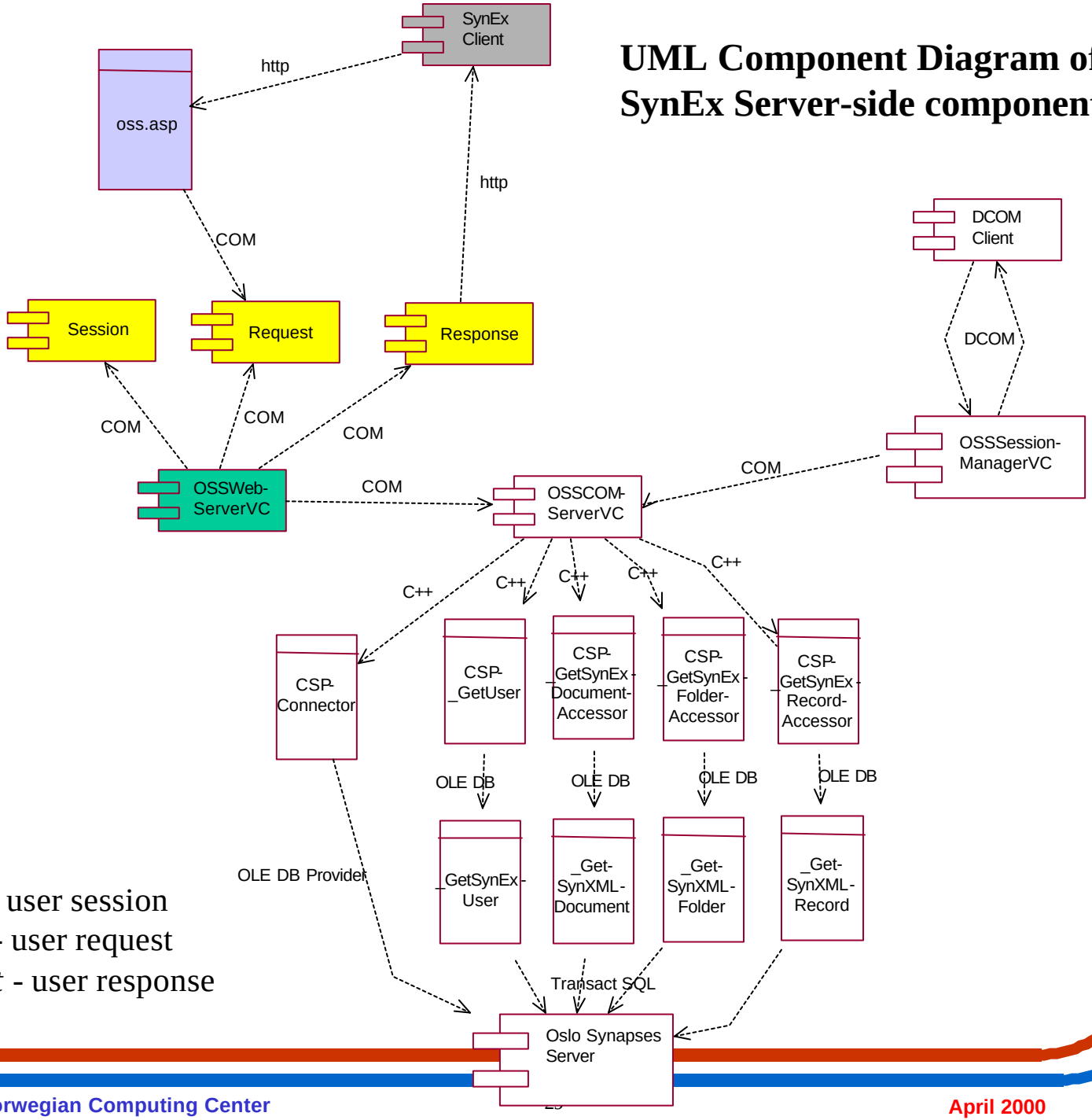
  Visual Basic, Visual C++, Visual J++ offer better development environments

```
<%@ Language=VBScript %>
<%
On Error Resume Next

Set objServer = Server.CreateObject("OSSSynExDemo.COSSASPServer")
objServer.HandleClientRequest()

If Err.Number <> 0 Then
   Response.Write(”…error message to client - e.g. XML formatted...")
   Err.Clear
End If
%>
```

UML Component Diagram of SynEx Server-side components



IIS COM objects:

- *Session object* - user session
- *Request object* - user request
- *Response object* - user response

Client-side Components

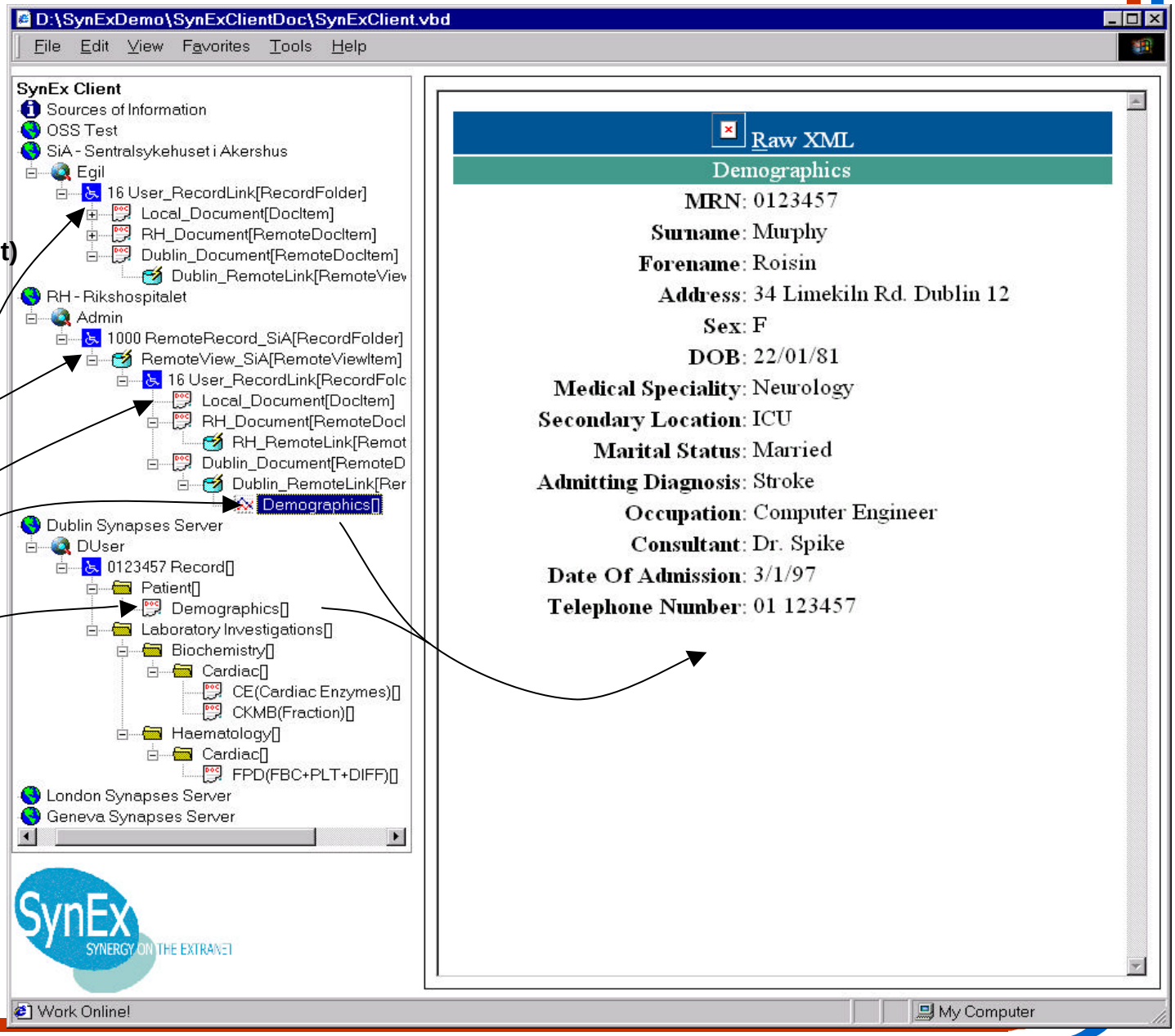
Example - Sharing Records at SiA, RH and Dublin

RH record (1000) references a SiA record (16) with a document (Dublin_Document) that is a reference (Dublin_RemoteLink) to a Dublin document (Demographics)

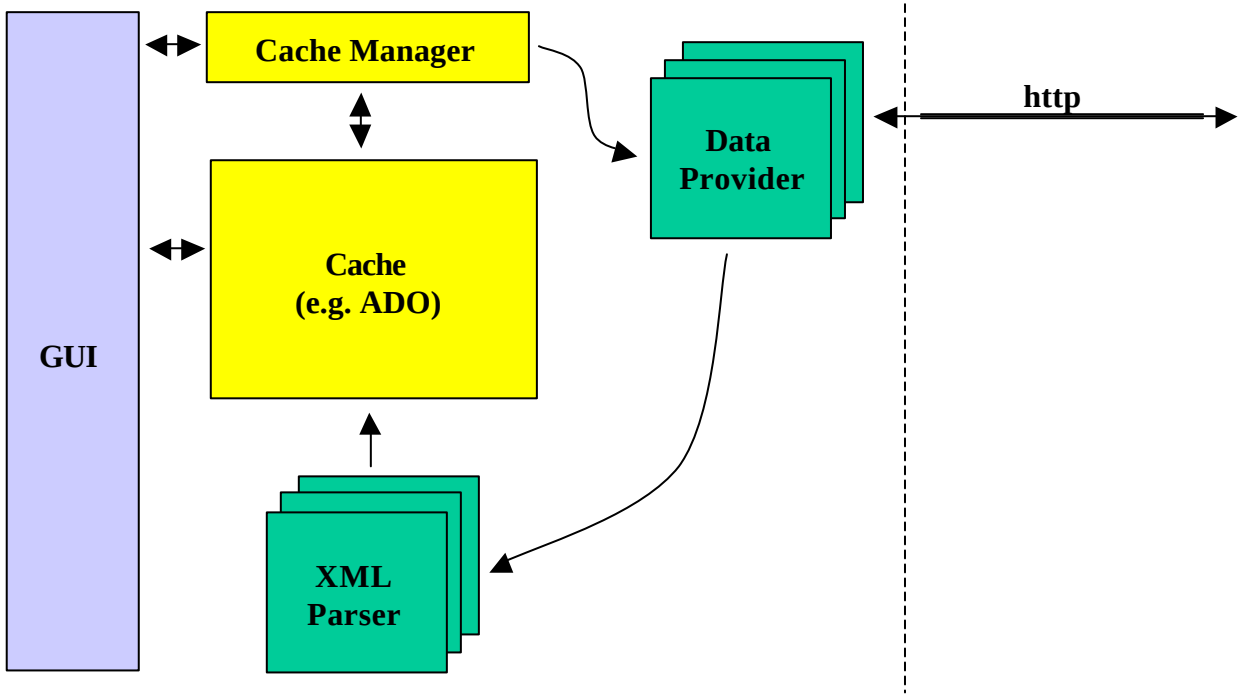
RH record

SiA record

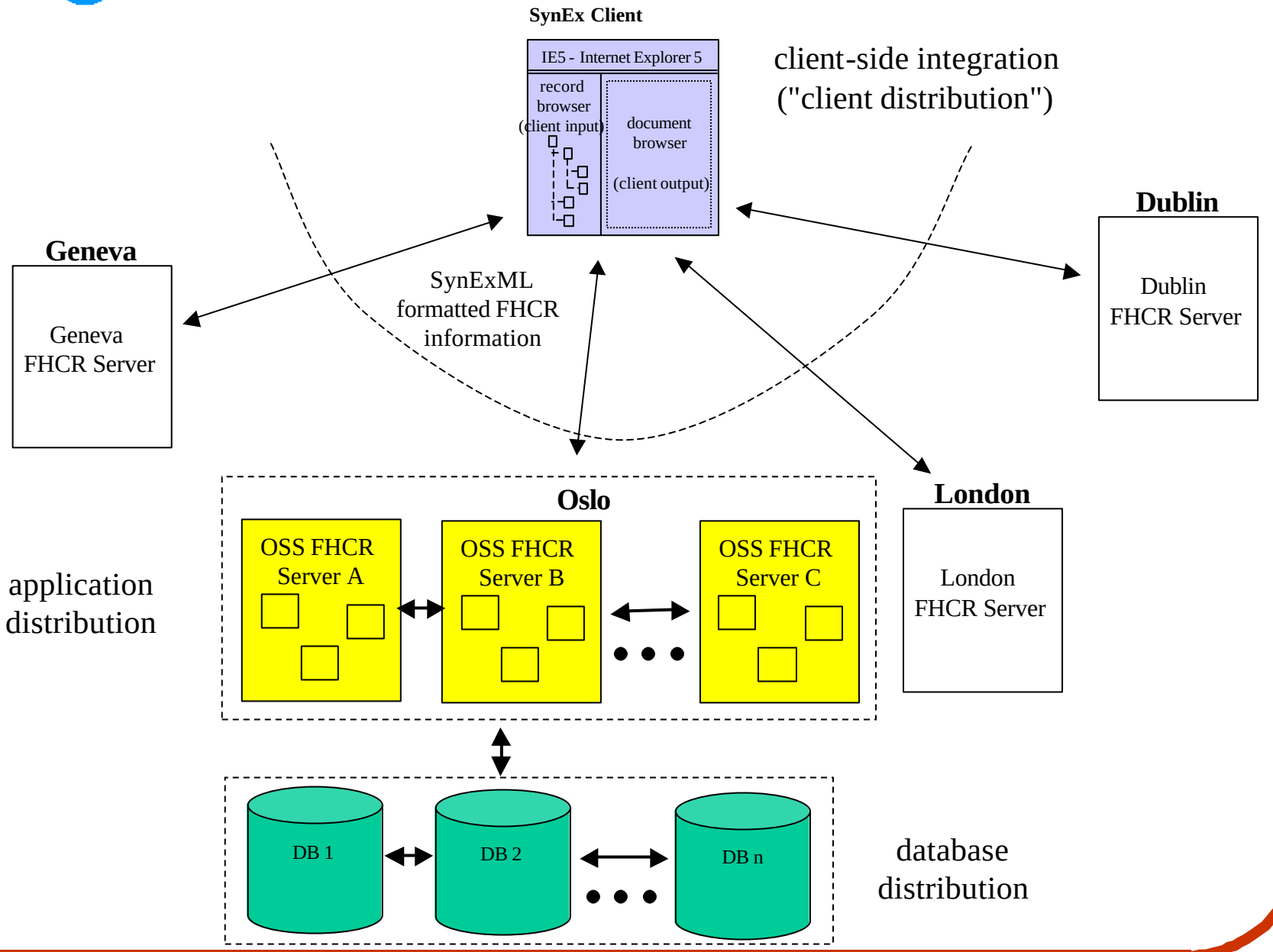
same Dublin document



(example) Client Architecture - "thick" clients



Client- vs Application- vs Database Distribution



Model and Meta-Information Management

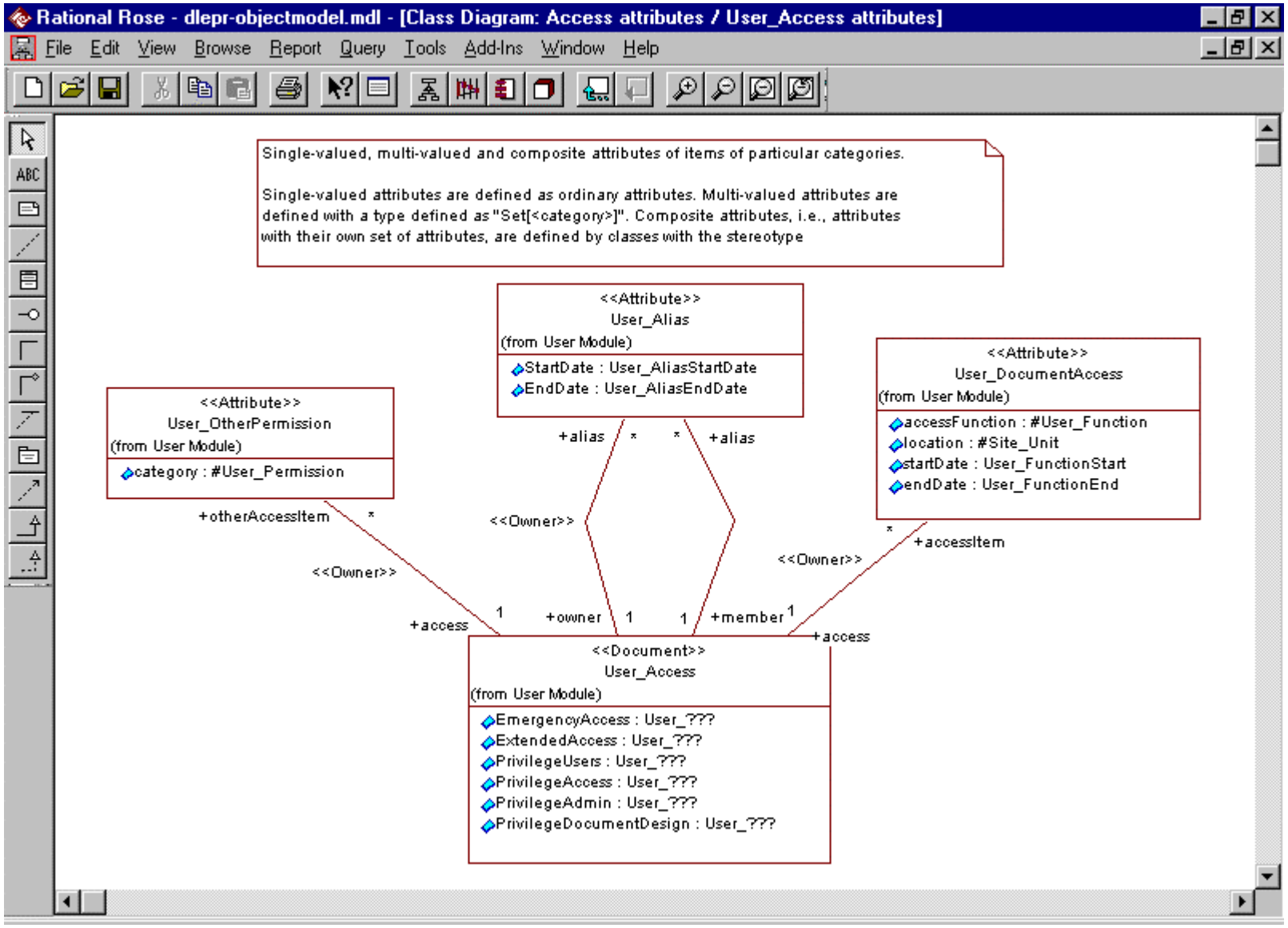
Characteristics of Rational Rose/UML

+ “Mainstream” - well-known and seen as a standard
+ Information modelling and explicit object interaction modelling
+ Object model available via COM/automation - it can be extended and customised
+ Code generation (but **not** production code…)
+ Informal (…this can be a plus)

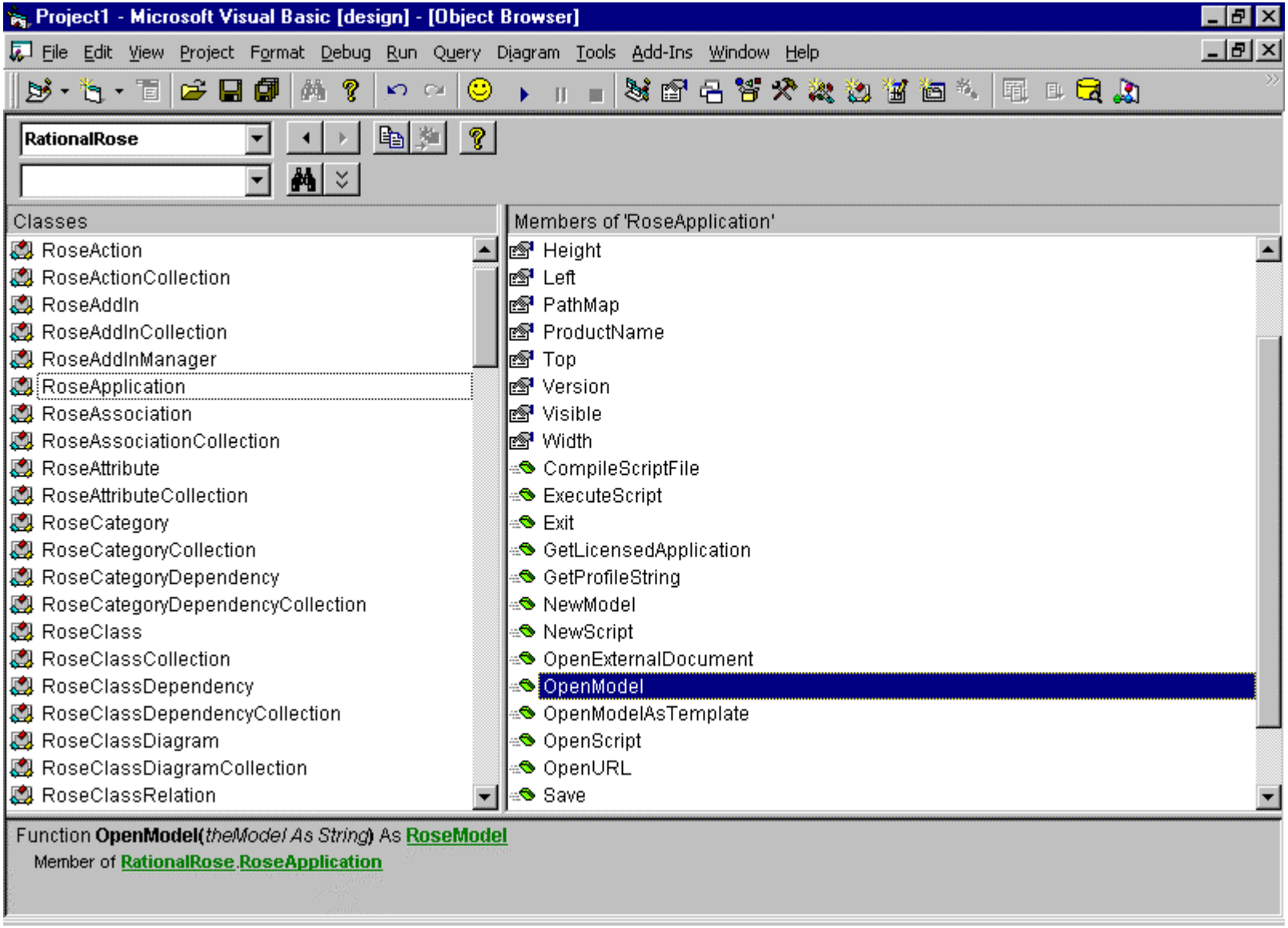
÷ Business rules and behaviour other than explicit object interaction
÷ Conceptual errors cannot be detected - models are not correct/incorrect - no modelling tool can distinguish good from bad models (and this is difficult also for experienced modellers)
÷ Incomplete
÷ Slightly confusing organization (at least at first…)

- Consider it mainly as a drawing tool and as a model repository
- Use only those parts that are well understood/agreed upon, and use it consistently - do **not** “over-model”
- Modelling syntax is not essential, but you are not likely to do e.g. Class Diagrams any better...
- Assuming that analysis/design is essential to large-scale software development, then a modelling tool can be useful to establish good routines for planning and documentation, and as a means for unambigous communication internally and externally.

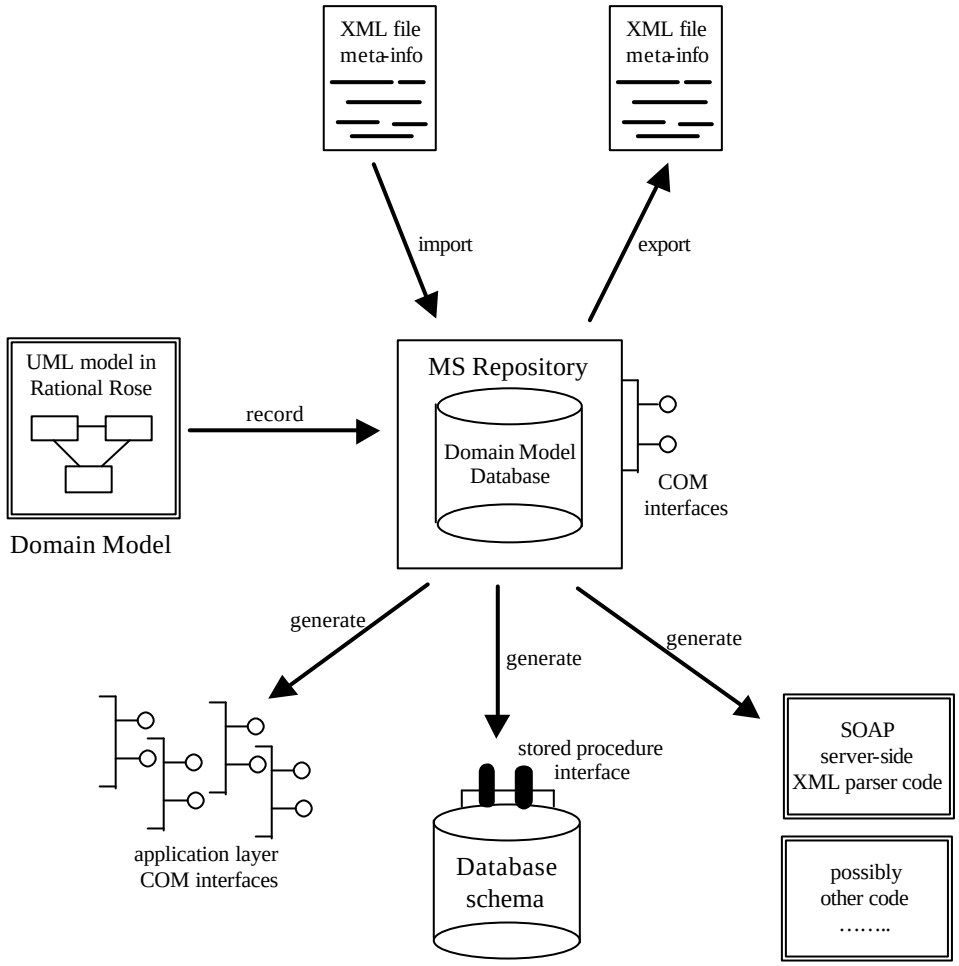
Example Rational Rose Class Diagram



The Rational Rose Object Model for COM/Automation



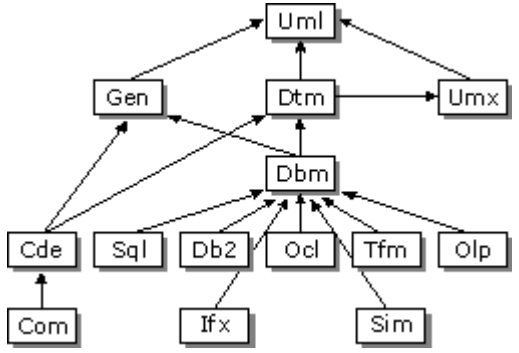
Generate Server-side Code from Meta-Information



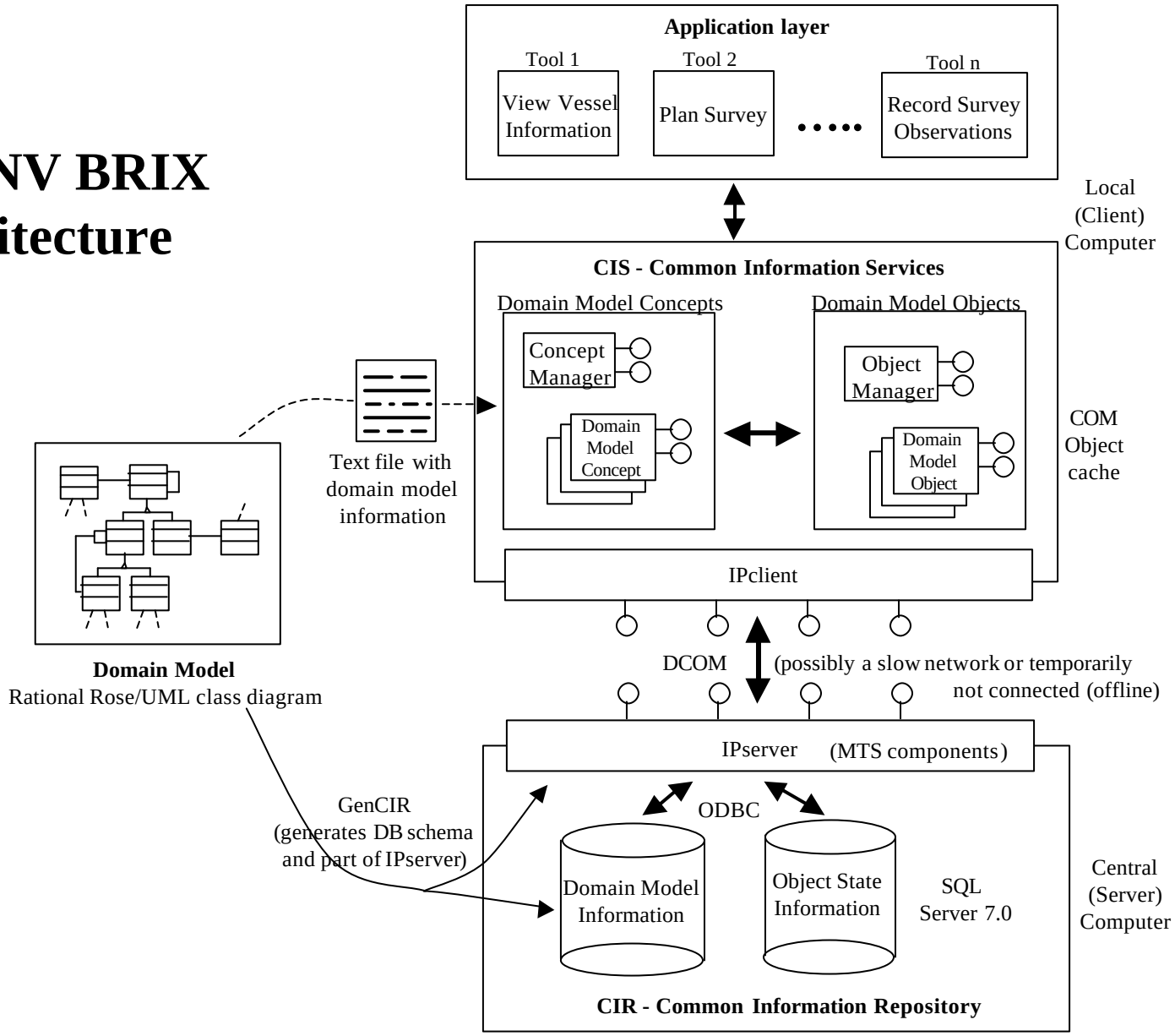
Microsoft Repository

- Meta-information management
- Object Information Model
- Extendable Subject Areas
- COM/Automation access

Subject areas in the current Open Information Model



The DNV BRIX Architecture



SINAI

Seamless Integration of Non-homogenous Applications and their Information

SINAI I - Utilizing Model and Meta-Information

Information models and object models play key roles in an information system architecture.

The goal of SINAI I is to create and maintain models in UML/Rational Rose that are used to partially generate domain-specific code in server- and client-side components;

e.g. DB schema, stored procedure interfaces, OLE DB wrapper classes, application layer IDL, XML/SOAP parsing, client cache's, and more.

Components at all layers can be specialised manually as required, e.g. for performance or the implementation of business rules that cannot be generated.

The use and format of the XML should be transparent to higher levels in the architecture.

- The server-to-client XML is formatted according to UML with information on objects instantiated from UML classes
- The client-to-server XML is formatted as object method calls

Semantically both a client and a server uses UML as the basis for their common understanding.

Clients will not only receive object information, but also UML meta-information on these objects; e.g. what are their interfaces, which methods to they offer, and more.

SINAI II - Generic and Extendable Information Systems

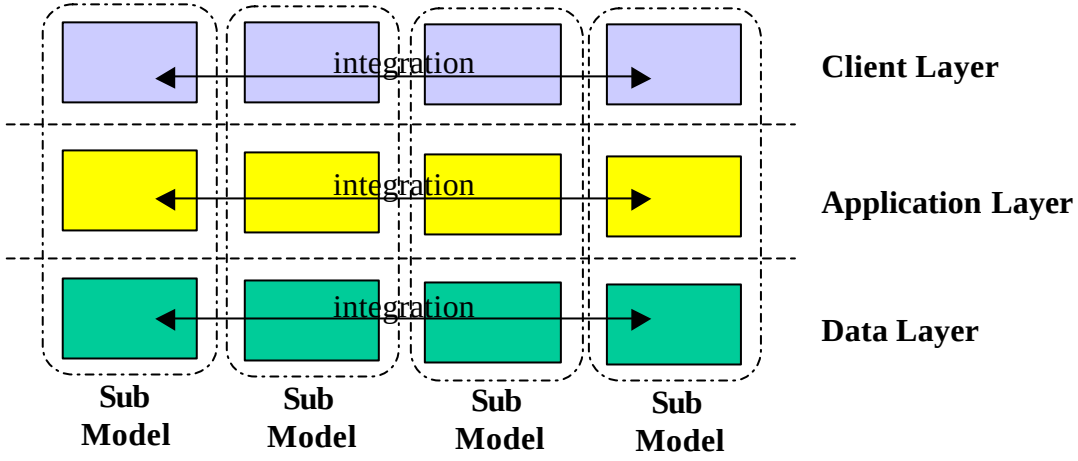
The kind of information that an information system must be able to manage may not be known at design-time

Possible solutions:

- A very generic database schema and application layer, with support for specialisation and customisation (Synapses)
- A very generic application layer that can be applied to "any" UML Class Diagram (BRIX)

Both these solutions have some disadvantages due to their generic nature.

Genericity provides flexibility - but at the cost of added complexity amongst others



SINAI II aims to provide an extendable solution without being generic, but instead by integrating "submodels" at all levels of a layered information system architecture.

Concluding Remarks

Concluding Remarks

Believe that component technology will become the core technology for most program developments. Exceptions may be very specialized, domain-specific platforms (e.g. certain mobile devices).

Benefits:

- Technical interoperability
- Programming language independence
- Location transparency (....not 100%....)

Component technology is a "natural" extension of object-oriented programming, but it may have an even more profound influence on programming than what object-oriented programming languages have had (if disregarding the fundamental conceptual understanding implied by OO as a methodology).

Component technology is still young - but there exist many business-critical applications and enterprise systems based on component technology.

PS: Learning how to develop components is more demanding than learning e.g. SQL or XML.

Concluding Remarks (cont.)

(D)COM versus CORBA/EJB/Java

- They are likely to coexist
- COM are sometimes preferred by commercial companies since there are many more COM components and COM-compatible tools available on the market
- Compatibility between COM components seem better than the compatibility of ORB's made by different vendors

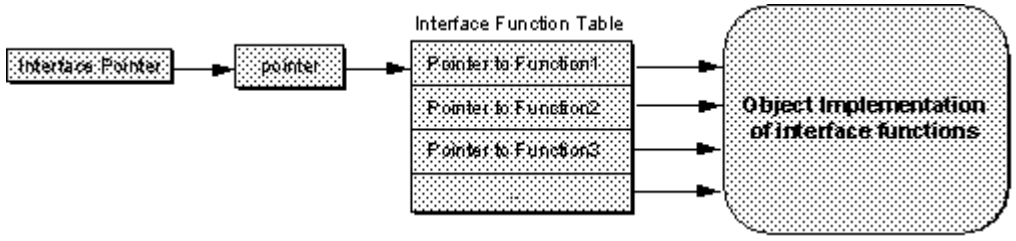
Component Technology and Programming Languages

Component technology may bring new life to specialized and "small" programming languages

On the other hand - a production environment where different components are implemented in different programming languages is not very manageable despite being technically interoperable - e.g. debugging a combination of Visual C++, Lisp, Object-Oriented Cobol, etc....

Etcetera - Technical Details

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++

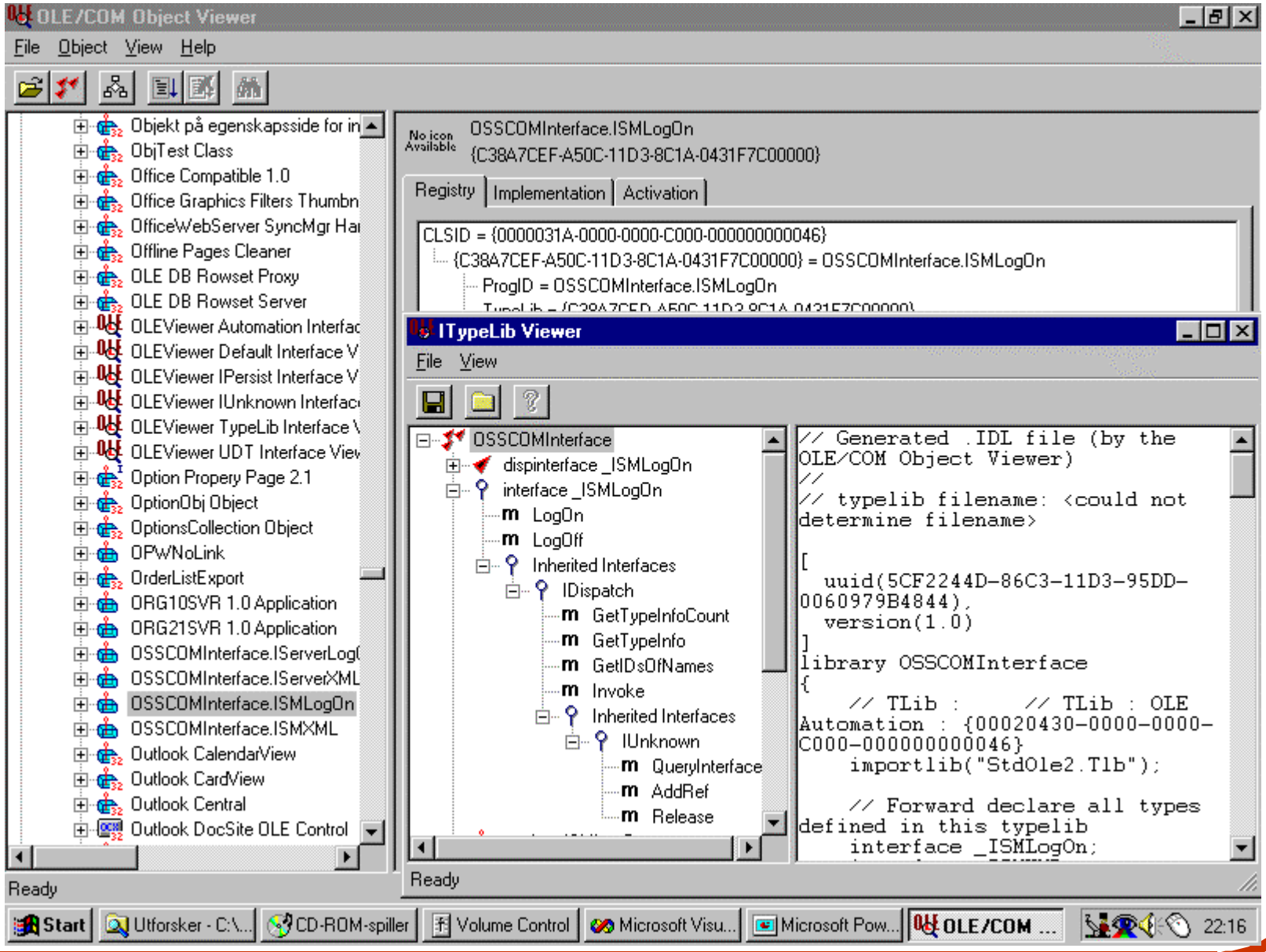
  A COM interface and its functions is similar to an abstract base class with a set of virtual functions in C++

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

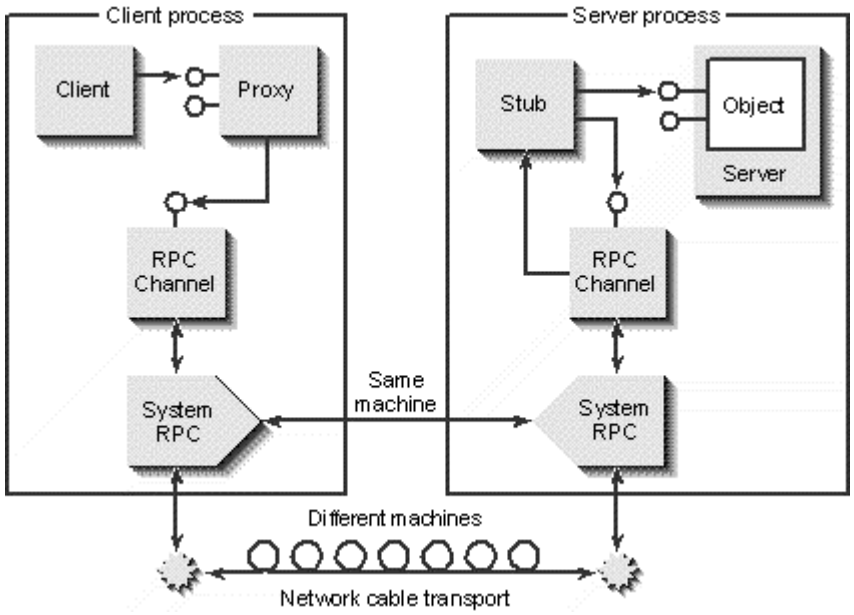
- **Dispatch interfaces** - query the interface for its functions and their signatures

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

Information on Interfaces and Components

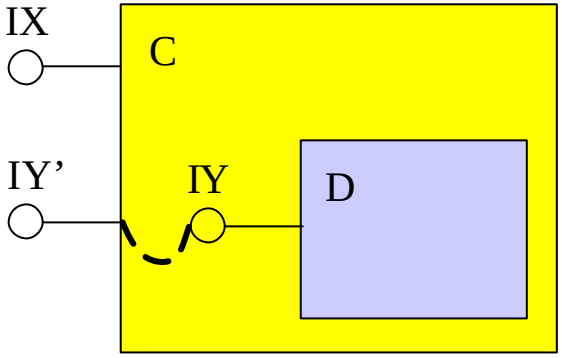


Marshalling for Out-of-Process Components



Integrating COM Components via Containment vs Aggregation

Containment



Aggregation