Web Services

Giuseppe Attardi
Università di Pisa

Overview

- From COMponents to .NET
- Web Services Architecture
- Demo
- Reflection and Metadata
- 2-Way Web
- Open Issues

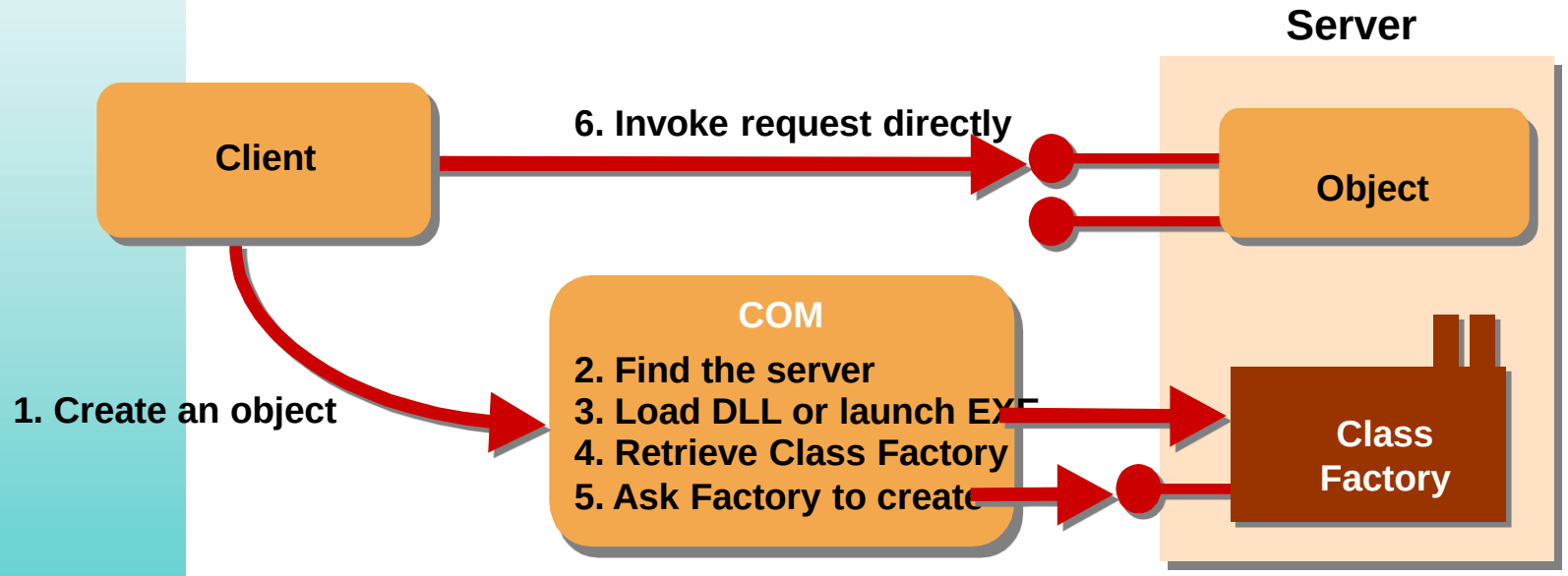
Software Components

COM Classes and Servers

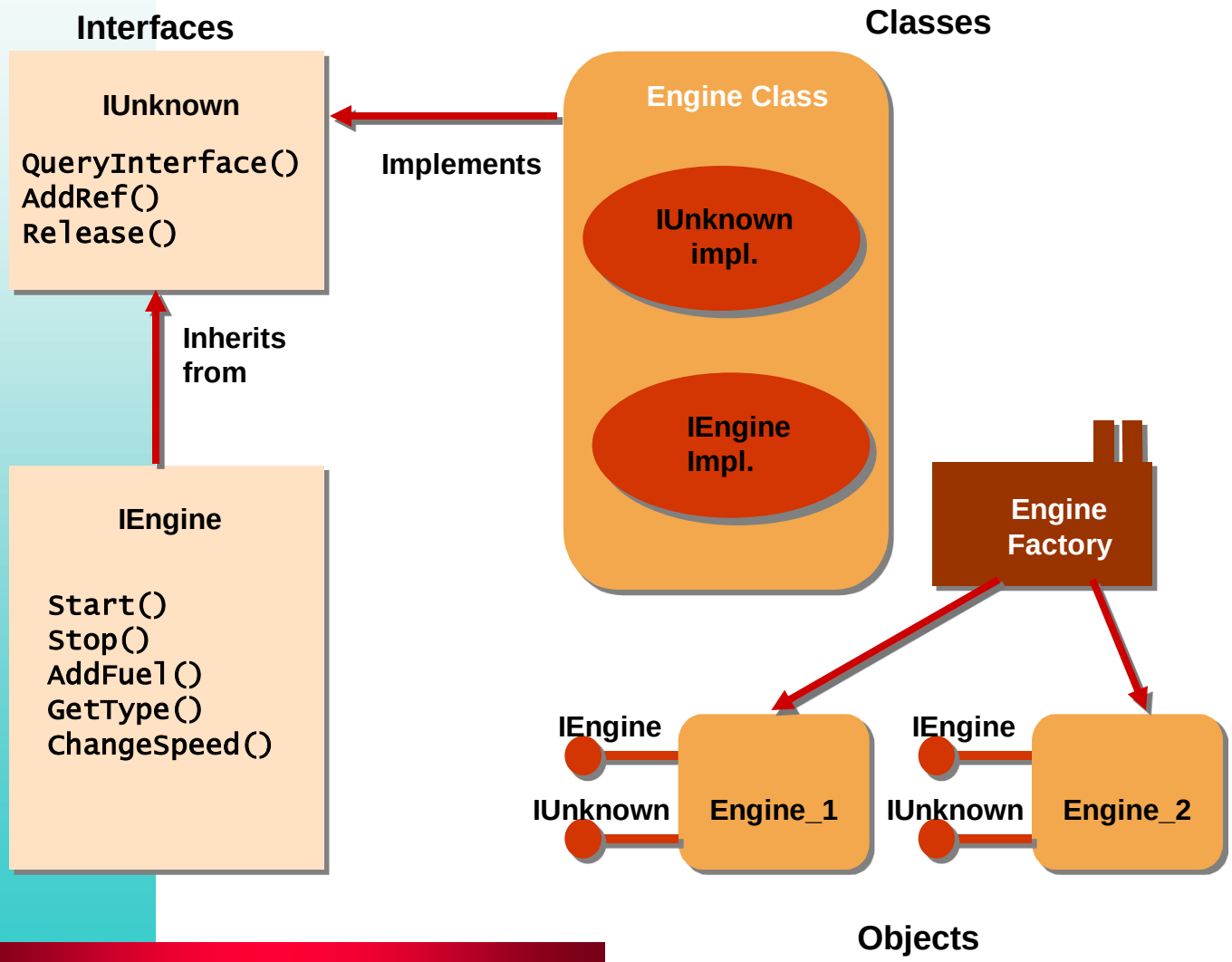
- **COM class: body of source code that implements COM interfaces**
- **provides real functions in any supported programming language for each interface method it supports**
- **each COM class has a unique identifier (CLSID)**
- **client asks COM to create an object and return interface pointer**
- **client applications interact with COM objects through interface pointers**

- client not dependent on implementation details of COM
- COM servers:
  - in-process server: DLL loaded into client process calls go directly to object created in the client's process
  - out-of-process server: separate executable, either on same machine as a client or on remote machine; calls go first to an in-process proxy which uses RPC; in the server, stub object receives each incoming call and dispatches to appropriate COM object
- ActiveX control is in-process COM server object

Object Creation



Interfaces, Classes and Factories



COM Interfaces

- **COM interface defines behavior or capabilities of software component as a set of methods and properties**
- **interface is contract that guarantees consistent semantics**
- **each COM object must support at least one interface (IUnknown)**

COM pros/cons

- **PROs**
  - **Access to OS functionality**
  - **Faster and easier to write apps**
  - **Third-party COM components**
- **CONs**
  - **Requires infrastructure and tools**
  - **Client/server kept separate (e.g. different strings implementations)**
  - **DLL hell**

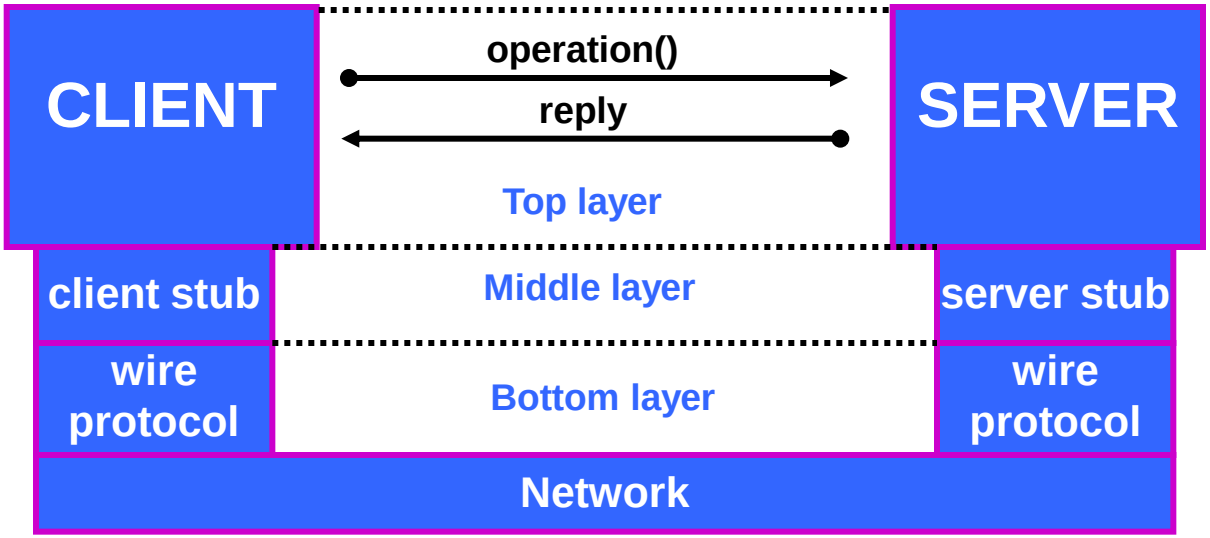
History of Distributed Object Models

Communication Protocol Models:

- **Message passing/queueing (DCE)**
- **Request/response (RPC)**

- **1980** **model based on network layer (NFS, DCE RPC)**
- **1990** **object-oriented RPC, to link objects**

Remote Procedure Call



ORPC

- **ORPC codify mappings between objects at language level**
- **Server-side middleware locate and instantiate object in target process**
- **Microsoft DCOM and CORBA IIOP were dominating ORPC protocols**

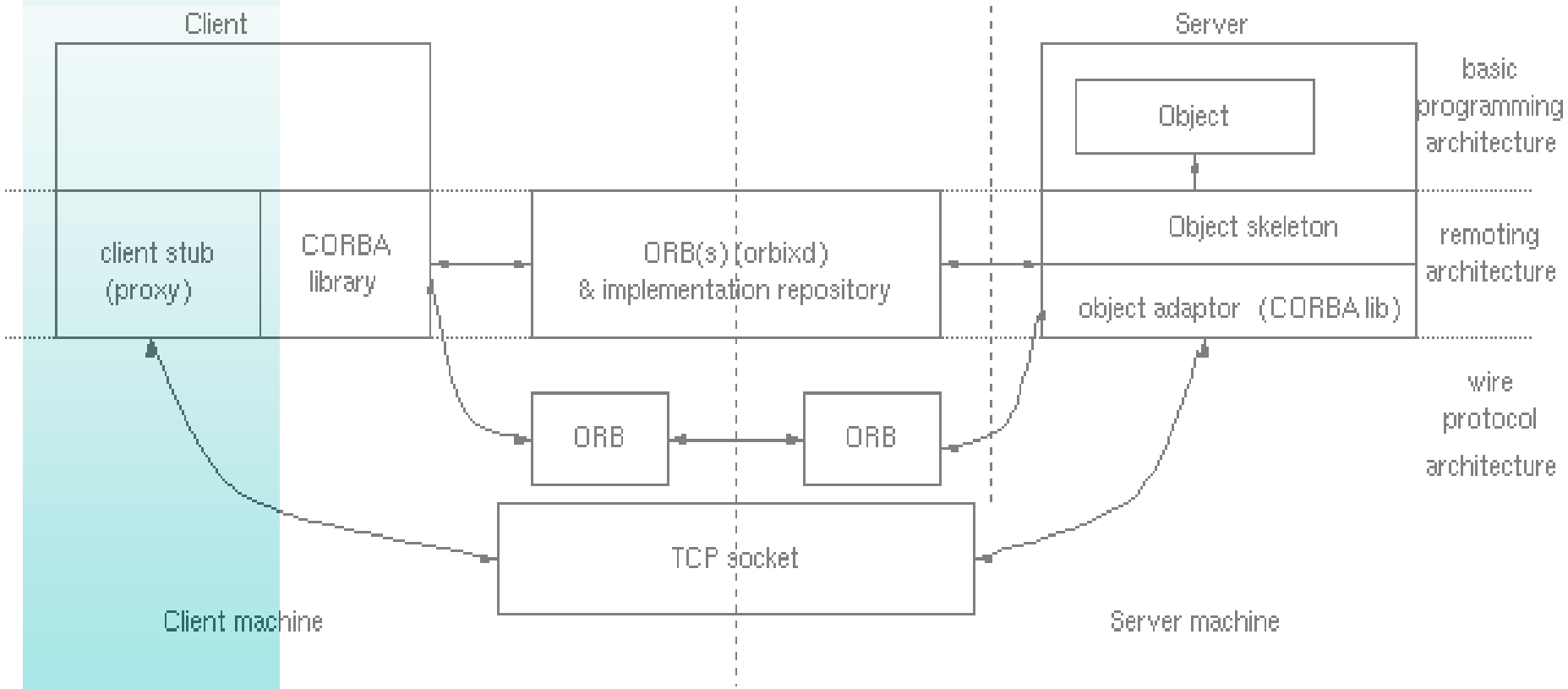
CORBA

- **OMG's specification for interoperability between distributed computing nodes**
- **Goal: heterogeneous environments communicating at the object level, regardless of implementation of endpoints**
- **ORB: middleware that establishes requestor-provider relationship**

ORB

- **Receives invocation message to invoke specified method for registered object**
- **Finds object, unmarshals parameters, invokes method, marshals and returns results**
- **Requester needs not to be aware of location, language or OS of object**

CORB Architecture



Interface Definition Language

- **Language neutral specification**

```
interface Polynomial : MathObject {
   sequence<Monomial> monomials;
   int rank;
   Polynomial add(in Polynomial p);
};
```

- **Mappings to several languages**
- **Tools (compilers) generate stubs and skeletons in various languages**

Note. No way to know at run-time which interfaces an objects provides: IDL gets compiled away

DCOM

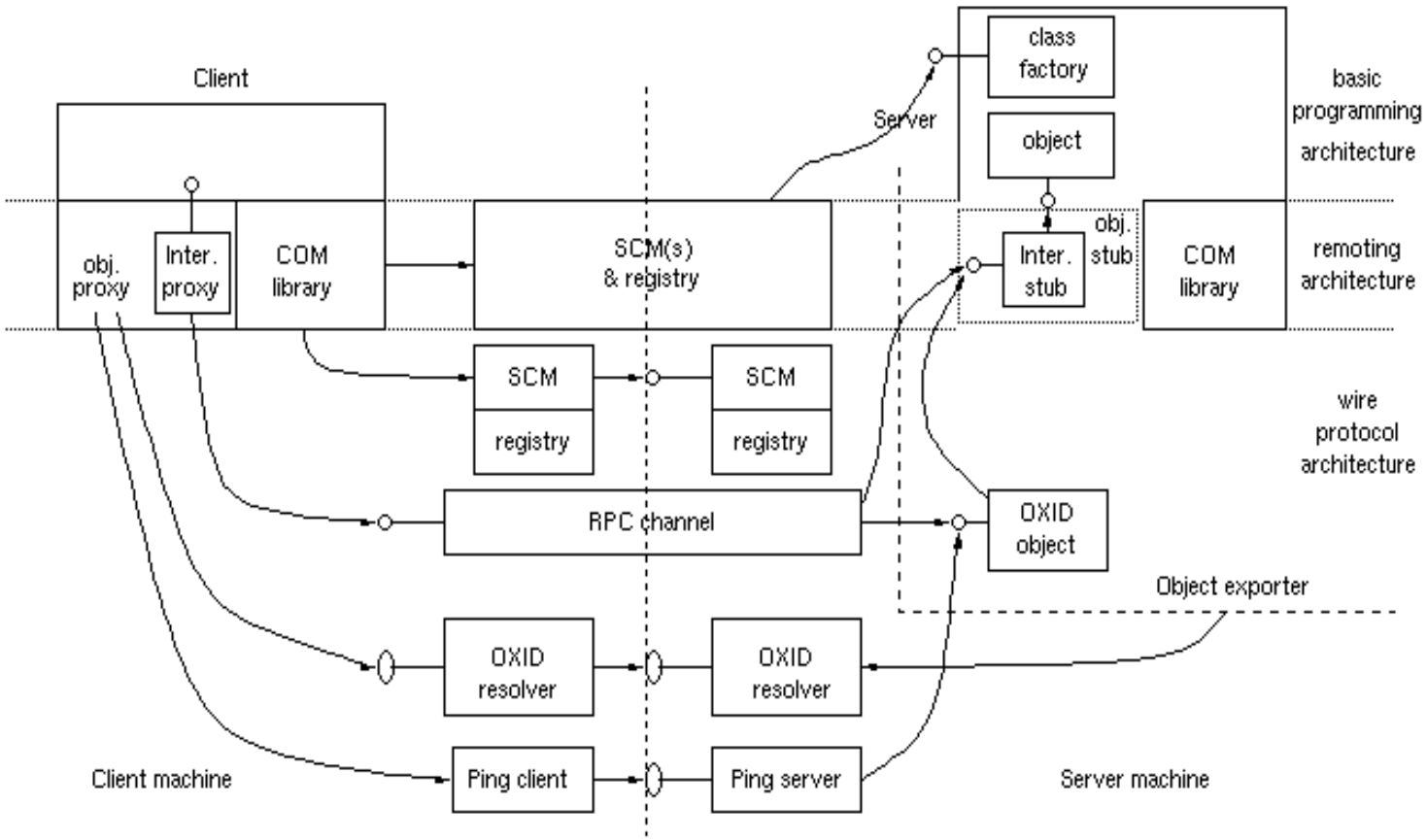
- DCOM distributed extension to COM
- builds an ORPC layer on top of DCE RPC
- COM server can create object instances of multiple object classes
- COM object supports multiple interfaces, representing different view or behavior of the object
- interface consists of a set of functionally related methods

DCOM Interfaces

- **interfaces described using MIDL**
- **MIDL compiler generates proxy and stub code in C or C++ from interface definition**
- **generated proxy code provides client-side API**
- **stub objects decode incoming client requests and deliver to appropriate object in the server**

- COM client interacts with COM object by acquiring a pointer to an object's interface and invoking methods through that pointer, as if the object resides in the client's address space
- interfaces follow standard memory layout, same as C++ vtable
- specification at binary level
- integration of binary components in different languages (C++, Java, Visual Basic)

DCOM Architecture



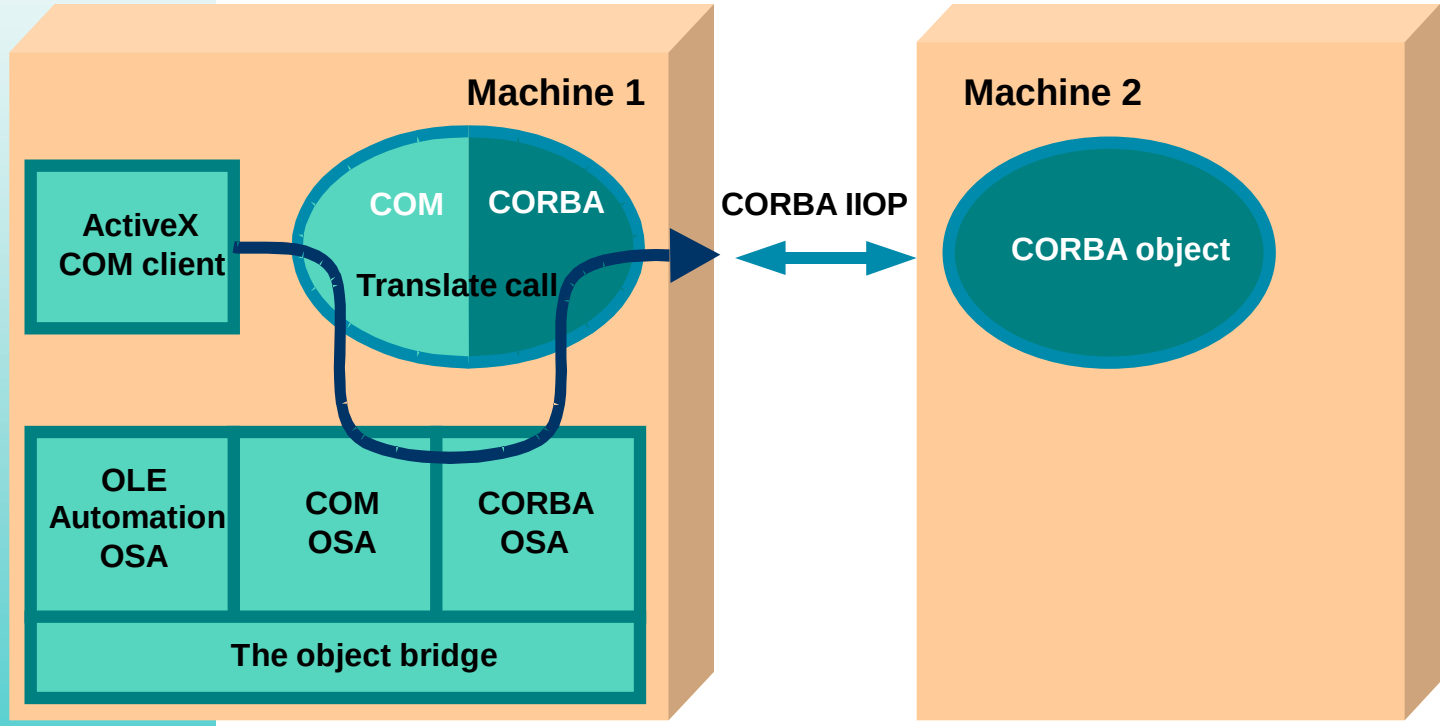
DCOM Overview

- **proxy and stub code interact with appropriate runtime libraries to exchange requests and responses**
- **each interface has UUID**
- **QueryInterface method of IUnknown**
- **QueryInterface returns an interface pointer**
- **interface pointer points to COM binary data structure**
- **client application must know CLSID and IID for an interface**
- **standard dictates interface functions calling conventions**

CORBA / COM interoperability

- **Naming of communication endpoints:**
  - **CORBA: Interoperable Object Reference**
  - **DCOM: OBJREFs (include reference counting)**
- **Support for multiple interfaces (only in DCOM)**
- **Format of payload parameter values:**
  - **DCOM: Network Data Representation**
  - **CORBA: Common Data Representation**

COM-CORBA Interoperation



CORBA and DCOM limitations

- **DCOM platform limitation**
- **CORBA, subtle incompatibilities require ORB from same vendor**
- **Reliance on closely administered environments**
  - **IIOP must cross firewalls**
- **Programming difficulties in data alignment and data types**

Quest for Net Objects

1993	**COM**
1996	**Java**
1997	**Mary Kirtland's articles in MS System Journal present first sketch (COM+)**
1997	**Sun vs Microsoft over Java licensing**
1999	**Java 1.2**
2000	**MS announces .NET, CLR, C#**

Web Computing

- **Programming with distributed components on the Web:**
  - **Heterogeneous**
  - **Distributed**
  - **Multi-language**

Beyond browsing

- **Access and act on information**
- **More control, better decision-making and easier collaboration**
- **Optimal support for different devices**
- **Open to partners: each can build its portion of the application**

Classes of Use

- **Web Services**
- **2-way Web**
  - **Full interactive capabilities of desktop applications**

Web Services

Web Service: Definitions

- **Component for Web Programming**
- **Self-contained, self-describing, modular component that can be published, located, and invoked across the Web**

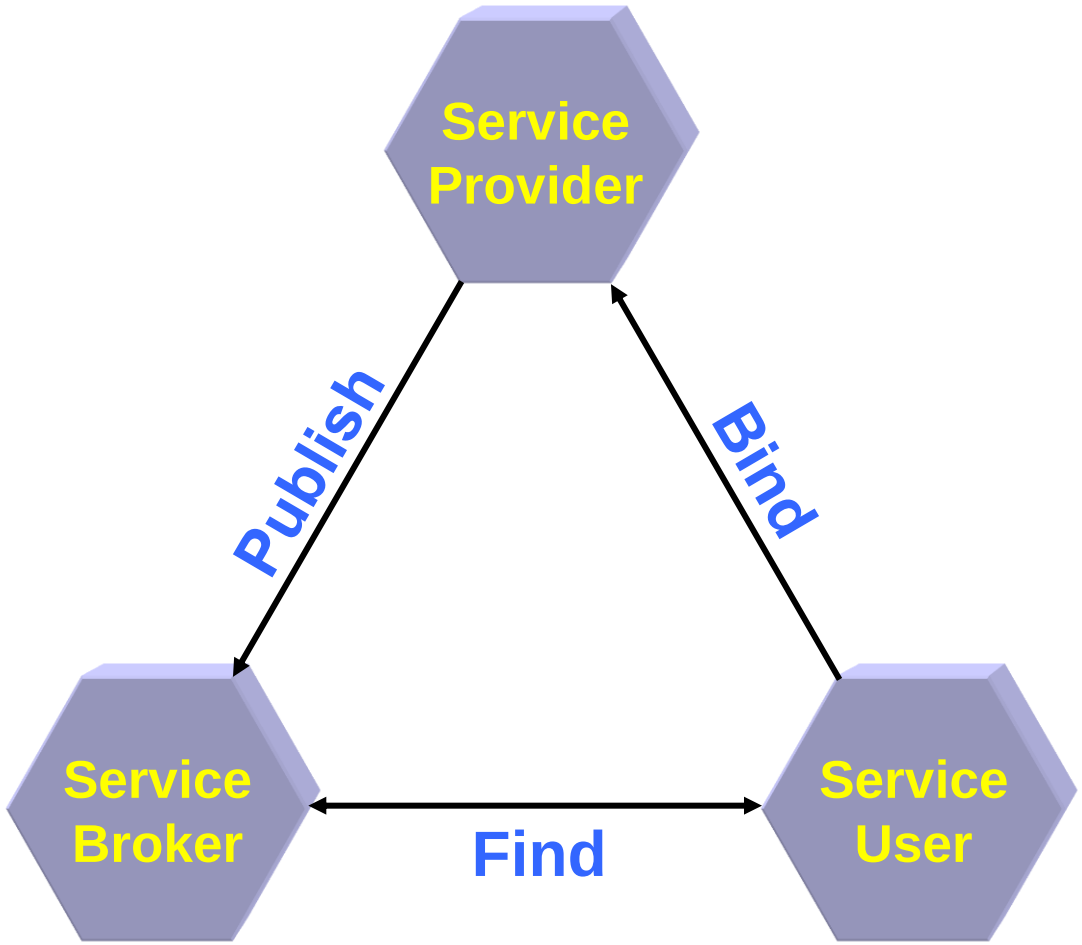
Web Services: Properties

- **can be used either internally or exposed externally over the Internet**
- **accessible through a standard interface**
- **allows heterogeneous systems to work together as a single web of computation**

Properties

- **Loosely coupled**
- **Ubiquitous communication**
- **Universal data format**

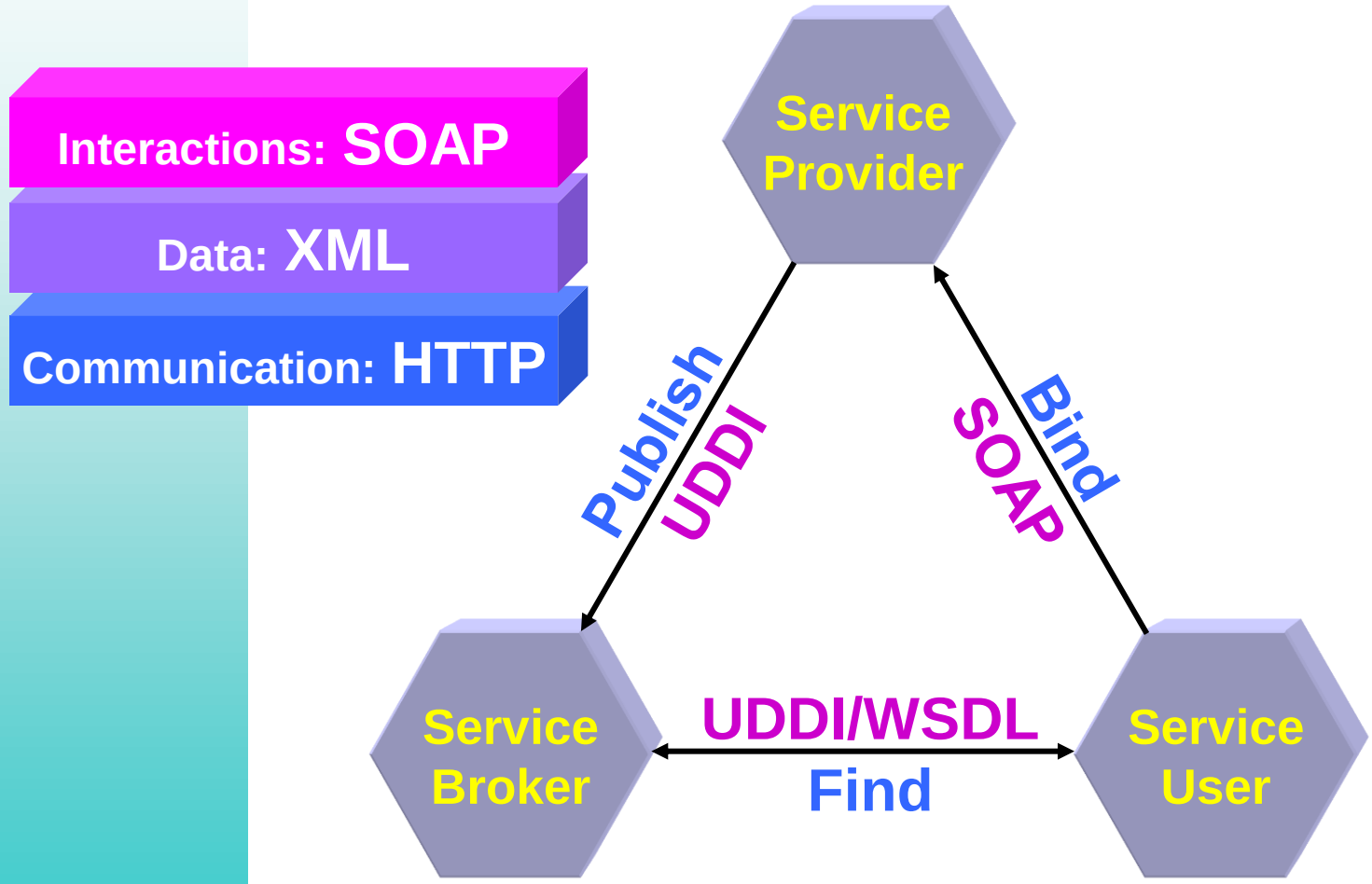
Service-Oriented Architecture



Web Service Scenario

- **Provider builds and defines the service in WSDL**
- **Provider registers the service in UDDI**
- **User finds the service by searching UDDI registry**
- **User application binds to the Web service and invokes its operations via SOAP**

Web Service Architecture



Web Services Protocols

Infrastructure Elements

Directories

central location to locate Web Services provided by other organizations (e.g. UDDI registry)

Discovery

locating WSDL for a particular Web Service

Description

defines what interactions the Web Service supports

Wire Formats

enable universal communication (e.g. SOAP)

SOAP

Wire-protocol based on XML and HTTP that consists of:

- **an envelope for describing what is in a message and how to process it**
- **a set of encoding rules for expressing instances of application-defined data types**
- **a convention for representing remote procedure calls and responses**

Sample SOAP request

```
POST /CurrencyServer/CurrencyExchange.asmx HTTP/1.1
Host: theseus
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: http://di.unipi.it/webservices/Euro

<?xml version="1.0" encoding="utf-8"?>
 <soap:Envelope
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:xsd=http://www.w3.org/2001/XMLSchema
  xmlns:soap=http://schemas.xmlsoap.org/soap/envelope>
 <soap:Body>
  <Euro xmlns=http://di.unipi.it/webservices>
   <currency>string</currency>
  </Euro>
 </soap:Body>
</soap:Envelope>
```

Sample SOAP reply

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xmlns:xsd=http://www.w3.org/2001/XMLSchema
  xmlns:soap=http://schemas.xmlsoap.org/soap/envelope>
 <soap:Body>
  <EuroResponse xmlns=http://di.unipi.it/webservices>
   <EuroResult>double</EuroResult>
  </EuroResponse>
 </soap:Body>
</soap:Envelope>
```

Other .NET Wire-Protocols

- **HTTP GET**
- **HTTP POST**
- **SMTP**
- **… customized**

Web Service Description Language

WSDL Structure

Types	**Data type definitions**
Message	**Signature of request and reply for each method (≈ IDL)**
Port Type	**<service, protocol> ⇨ operations**
Operation	**method ⇔ messages**
Binding	**Protocol and data-format specification**
Service	**{ Port ⇔ binding }**
Port	**Address (≈ URL)**

WSDL example

- **Currency Exchange Service**
- **Methods**

```
double Rate(String From, String To)
double Euro(String Currency)
```

- **Service URL**

  http://theseus/CurrencyServer/CurrencyExchange.asmx

WSDL example

```
<message name="RateSoapIn">
 <part name="parameters" element="s0:Rate" />
 </message>
 <message name="RateSoapOut">
 <part name="parameters" element="s0:RateResponse" />
 </message>
 <message name="EuroSoapIn">
 <part name="parameters" element="s0:Euro" />
</message>
<message name="EuroSoapOut">
 <part name="parameters" element="s0:EuroResponse" />
 </message>
 <message name="RateHttpGetIn">
 <part name="from" type="s:string" />
 <part name="to" type="s:string" />
 </message>
 <message name="RateHttpGetOut">
 <part name="Body" element="s0:double" />
</message>
<message name="EuroHttpGetIn">
 <part name="currency" type="s:string" />
</message>
<message name="EuroHttpGetOut">
 <part name="Body" element="s0:double" />
</message>
<message name="RateHttpPostIn">
 <part name="from" type="s:string" />
 <part name="to" type="s:string" />
</message>
<message name="RateHttpPostOut">
 <part name="Body" element="s0:double" />
</message>
<message name="EuroHttpPostIn">
 <part name="currency" type="s:string" />
</message>
<message name="EuroHttpPostOut">
 <part name="Body" element="s0:double" />
</message>
```

. . .

Building A Server

- **Simplicity**
  - **Source file (plain text = notepad accessible)**
  - **Compiled at run-time similar to ASP.NET pages**
  - **Just hit save**
  - **File extension is `.asmx`**
- **File can be inline or in separate assembly**

Building TRUST

- **CLR exposes its elements**
- **Users can create elements directly**
- **Even when using tools, you can look at their output and change it**

Building A Server

- `<%@ WebService class="[class]" %>`
  - Names the class and/or language used
- `using System.Web.Services;`
  - Required namespace
- `[WebMethod]`
  - Method is ‘web callable’
- **Optional: WebService base class**
  - Access ASP.NET intrinsics


demo

Creating Web Service Clients

1. **Grab WSDL from Web Service**
2. **Create proxy from WSDL**
3. **Execute methods against proxy, passing input parameters**
   - **Proxy calls Web Service on your behalf**
   - **Web Service returns results to proxy**
4. **Retrieve results from proxy**
5. **Display results**

ASP.NET Client

- **Uses proxy and SOAP protocol to communicate with Web Service**
- **Steps:**
  1. **Use wsdl utility to create local proxy**
  2. **Compile proxy using vbc or csc**
     - **Place compiled proxy assembly in bin folder of Web**
  3. **Create client**
     - **Import proxy namespace, code to proxy's properties and methods**

demo

Using SOAP toolkit (no .NET)

```
// allocate a new SoapClient pointer
 m_pClient = new ISOAPClient;

// create the SoapClient pointer
m_pClient->CreateDispatch("MSSOAP.SoapClient")

// initialize it
m_pClient->mssoapinit("http://www.MyService.com/Calc.wsdl",
   "Calc", "CalcPortType", NULL);

// perform addition
double ISOAPClient::Add(double dblA, double dblB, DISPID
   dispid)
 {
   double result;
   static BYTE parms[] = VTS_R8 VTS_R8;
   InvokeHelper(dispid, DISPATCH_METHOD, VT_R8,
   (void*)&result, parms, dblA, dblB);
   return result;
 }
```

Client Side?

- **Use HTTP GET**
  - **handle XML yourself**
- **Use wsdl.exe, generate client-side program, invoke it through Jscript**
- **Use ActiveX or Java**

Microsoft .NET

- **A software platform for XML Web Services**

Personal Remarks

- .NET provides plumbing for interesting new developments
- Opportunity for experimenting with new programming languages
- We can start asking questions:
  - what new higher-level facilities can be designed
  - how can we contribute: improvements, extensions, applications
- Not PDC Questions:
  - when it will be available?
  - Will it have this feature?

Commercial Offerings

.NET http://microsoft.com/net

WebSphere
http://www.ibm.com/websphere

J2EE http://java.sun.com/j2ee

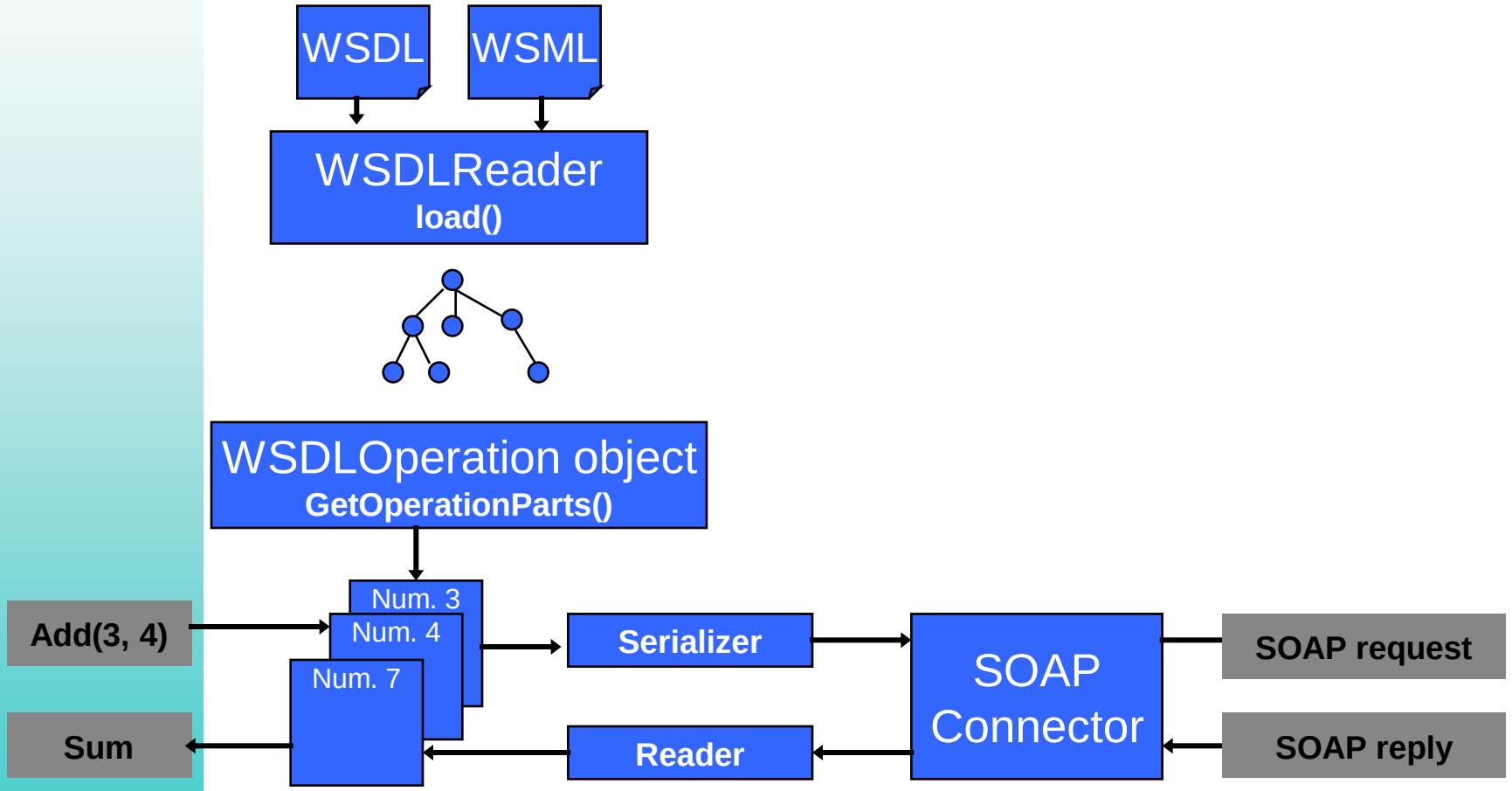
Role of CLR

- **Robustness**
  - **more and more programs run on server**
  - **avoid memory leaks**
- **Simplifies programming**
  - **Avoid burden of reference counting**
- **Reduce incompatibilities**
  - **Objects are remotely accessible**
  - **More easily reproduced if built on same basic elements**

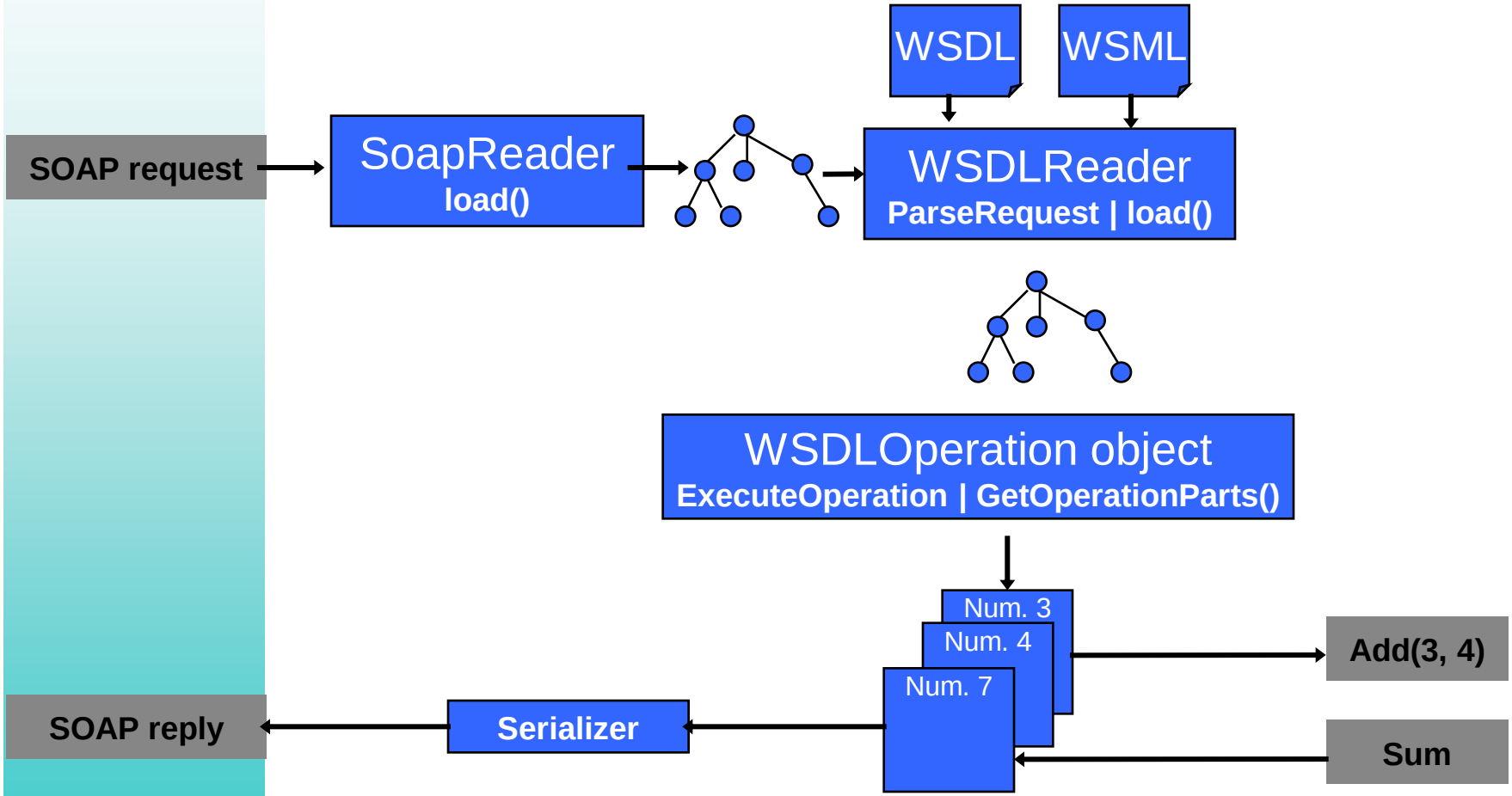
Reflection

- **Avoids IDL**
  - **Slight incompatibilities in CORBA**
- **Avoids type libraries**
- **Provides for dynamic invocation**
- **Allows customization**
  - **e.g. serialization**

Processing inside SOAP Client



Server-side SOAP



Reflection in Web Services

- **SOAP proxy performs:**
  - `Invoke(m, new object[] {arg1, arg2});`
- **SOAP message dispatcher:**
  - **parses request**
  - **creates parameter objects**
  - **determines object requested**
  - **instantiates object**
  - **gets requested method**
  - **invokes method with built parameters**

Reflection: Apache SOAP

- **SOAP requests addressed to:**

  server:8080//soap/servlet/rpcrouter/method

- **Servlet performs:**
  - `Call c = extractCallFromEnvelope(ServiceManager sm, Envelope e, SOAPContext ctx);`
  - `Response invoke(DeploymentDescriptor dd, Call c, Object o, SOAPContext reqCtx, SOAPContext resCtx)`

```
{ …
  m = MethodUtils.getMethod(o, call.getMethodName(),
    argTypes);
  return
     new Bean(m.getReturnType(), m.invoke(o, args));
  }
```

Meta Data

- **Reflection extracts metadata (no need for separate type library)**
- **Attributes: turned into metadata stored within IL**
- **Metadata accessible at runtime**
- **New attributes can be defined, for program use**

```
int (*fun)(int, char);

fun f = someFunction;

f(3,’a’);
someFunction(3,’a’);
apply(fun f, void* args) {
   f(args);
switch (args.lentgh) {
case 1: return f(args[0]);
case 2: return f(args[0], args[1]);
…
}
```

Use of Reflection

- **Building a high performance search engine**
- **Need way to store and retrieve objects from relational table**
- **Need reflection to serialize objects**
- **Solution before .NET: template metaprogramming**

Deployment Scenario

Customer performs title search

Music Portal

Service Broker

DJInterpool

DeadVinyl

FunkFactory

Portal invokes search Web Service

Service Broker invokes search services

Services respond with search results

Dynamic Objects (.NET)

- **Dynamic class creation**
- **Dynamic class loading**
- **Needed for interactive SQL interpreter**

Two-Way Web

Current Web Limitations

- **Thin but weak:**
  - **Not real-time**
  - **Not productive**
  - **Not interactive**
- **Client cannot initiate actions**
- **Browser pull: waste bandwidth**
- **Java, ActiveX: maintainability and security restrictions**

Current SOAPs

- **SOAP 1.1 specifications**
- **Implementations:**
  - **MS SOAP Toolkit 2.0**
  - **Apache SOAP 2.2 ⇨ AXIS (= SOAP 3.0)**
  - **SOAP::Lite for Perl**
  - **pocketSOAP**
  - **GSoap for C++**
- **Apache and MS are working on incompatibilities**

Conclusions

- **Web Services are quite promising**
- **Still missing:**
  - **some plumbing, interoperability**
  - **2-way interactivity**
  - **unified multistage programming**
- **Issues:**
  - **no more DLL Hell**
  - **but maybe, Namespace Hell**

References

Standards

SOAP http://www.w3.org/TR/SOAP

WSDL http://www.w3.org/TR/WSDL

UDDI http://www.uddi.org

Papers

1. Close up on .NET, DNJ, http://www.dnjonline.com/articles/essentials/iss24_essentials.html
2. M. Kirtland, Object-Oriented Software Development Made Simple with COM+ Runtime Services, MSDJ, http://www.microsoft.com/msj/defaultframe.asp?page=/msj/1197/complus.htm&nav=/msj/1197/newnav.htm