Optimization Services (OS) Framework

Robert Fourer and Jun Ma
Northwestern University

Kipp Martin
University of Chicago

November 15, 2005

Outline

Optimization Services (OS)

Service Oriented Architectures and Web Services

Optimization As a Web Service

OS Protocols
- OSProtocols: Representation
- OSProtocols: Communication
- OSProtocols: Registry

Solver Service – An OS Implementation

Client Service – An OS Implementation

Summary

Optimization Services (OS)

Software as a service! In industry, CRM (customer relationship software), tax preparation, Office Live, etc. are all becoming services. All of the major players in software are promising software as a service. There clearly is a trend away from the fat client loaded with lots of heavyweight applications.

So why not Optimization Services?

What is Optimization Services?

Short Answer: *A service oriented architecture for optimization based on Web services.*

Optimization Services (OS)

Optimization Services Long Answer:

- *A set of standards to facilitate communication between modeling languages, solvers, problem analyzers, simulation engines, and registry and discovery services in a distributed computing environment.*

Optimization Services (OS)

Optimization Services Long Answer:

- *A set of standards to facilitate communication between modeling languages, solvers, problem analyzers, simulation engines, and registry and discovery services in a distributed computing environment.*

- *These standards should be programming language, operating system, and hardware independent.*

Optimization Services (OS)

Optimization Services Long Answer:

- *A set of standards to facilitate communication between modeling languages, solvers, problem analyzers, simulation engines, and registry and discovery services in a distributed computing environment.*
- *These standards should be programming language, operating system, and hardware independent.*
- *These standards should be open and available for everyone in the OR community to use free of charge.*

Optimization Services

Optimization services is needed because there are:

- Numerous modeling languages each with their own format for storing the underlying model.
- Numerous solvers each with their own application program interface (API). There is no standard API.
- Numerous operating system, hardware, and programming language combination. It is difficult for software vendors to support every platform.
- No standard for representing problem instances, especially nonlinear optimization instances.
- No real standard for registry and discovery services.

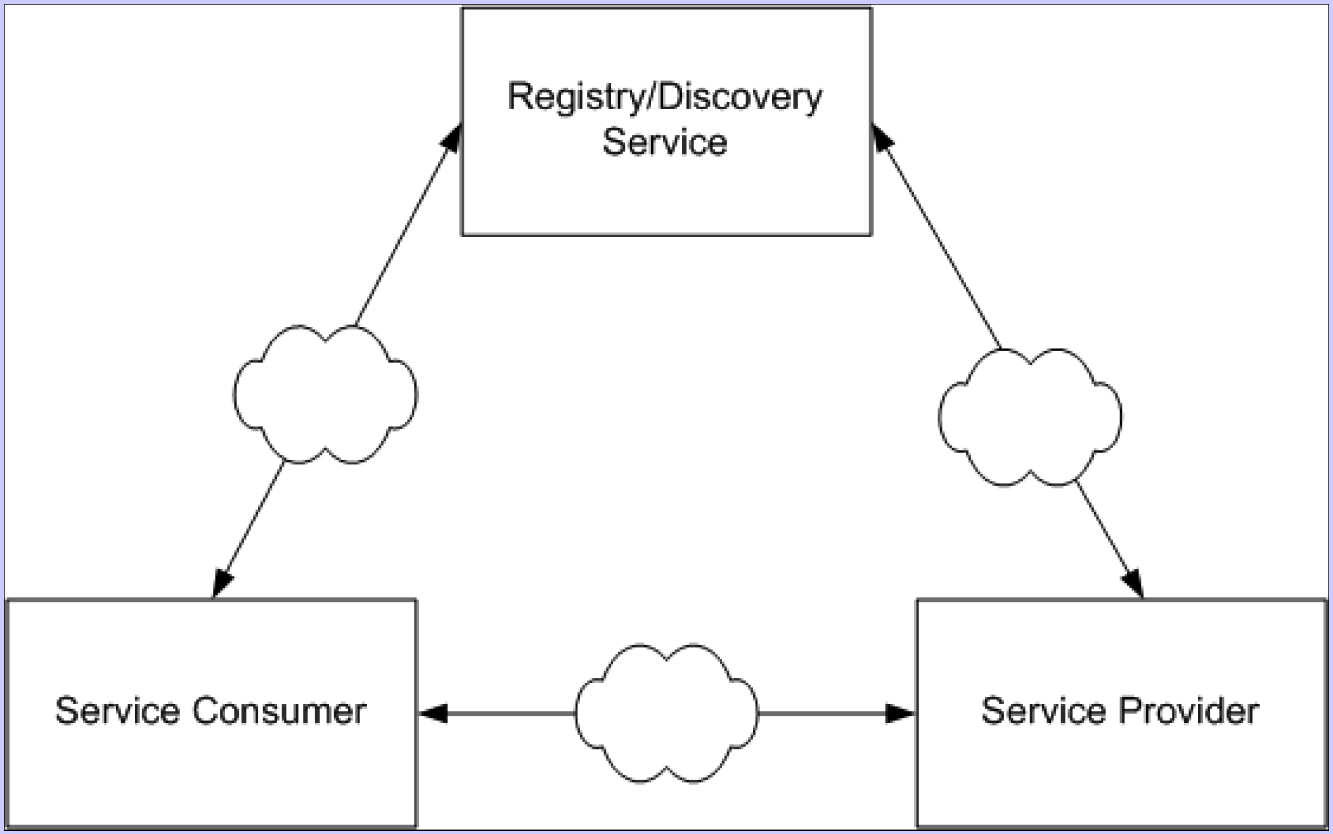
Optimization Services

In the rest of the talk we describe how to blend together:

- A Service Oriented Architecture (SOA) using Web Services
- Optimization Service Protocols – one way to view optimization systems is as a set of protocols
- Solver and Client Service implementations based on Web Services and OS Protocols

Service Oriented Architectures

Key Trend: An important trend in industry is the move to services oriented architectures and Web services. **All** of the major players such as IBM, Microsoft, Oracle, Sun, etc are talking about service oriented architectures and bringing out products.



Service Oriented Architectures

An SOA is a **philosophy** for how a distributed component architecture should work it is not a specific technology

Web Services is a technology that implements this philosophy.

Definition: Web Services is SOAP over a transport protocol such HTTP, SMTP, FTP, etc.

HTTP is the most common protocol and is the protocol we use HTTP in our implementations.

Web Services

Web Services is Popular because:

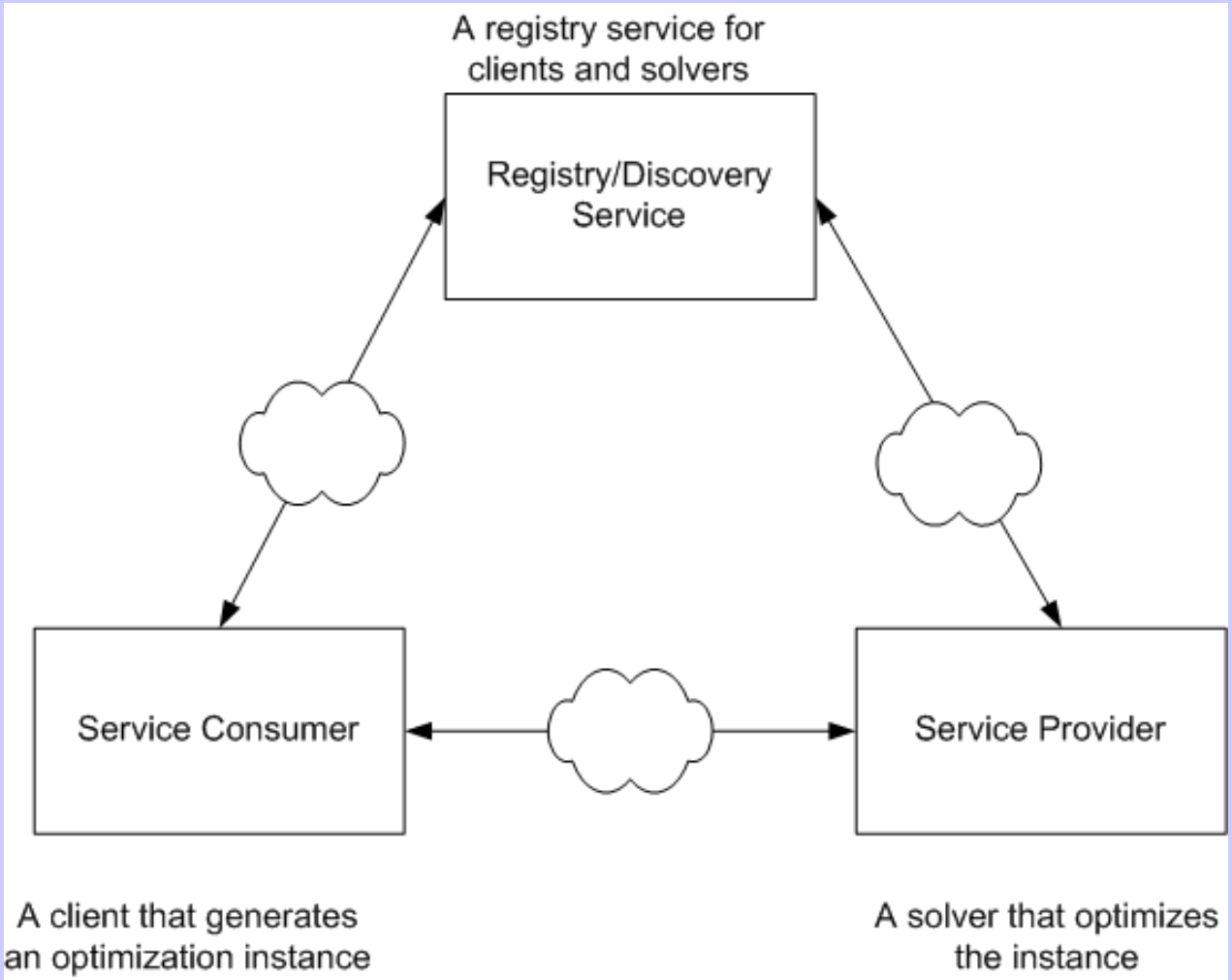
- Uses open standards, e.g. HTTP, XML, SOAP
- Can be used to develop rich clients
- Can be used by components people not necessary

Web Services makes use of three major protocols:

- SOAP (Simple Object Access Protocol)
- WSDL (Web Services Discovery Language)
- UDDI (Universal Description, Discovery, and Integration)

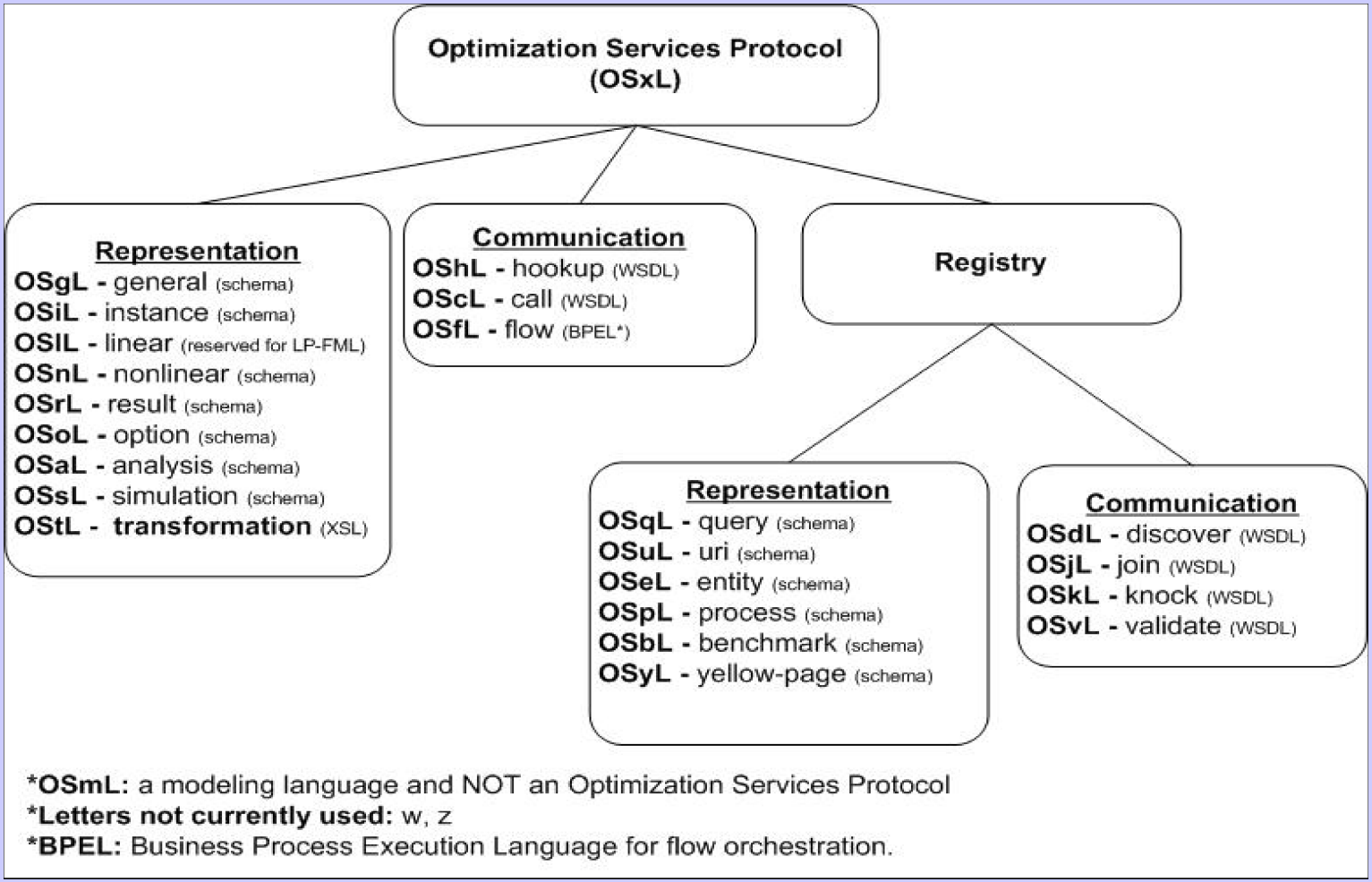
Optimization As a Web Service

Optimization Services: a service oriented architecture for optimization using Web services (SOAP over HTTP)



OSProtocols

Optimization Services: A set of protocols for **representation**, **communication**, and **registry**.



OSProtocols: Representation

The representation protocols are standards based on XML schemas. Several of the key representation standards include:

- **OSiL:** Optimization Services Instance Language. This is perhaps the most important representation protocol. OSiL is a standard for representing a very broad class of optimization problem instances. This standard is critical for exchanging problem instances between a solver (server) and a modeling language (client).
- **OSnL:** General nonlinear terms are handled separately, by defining an OSnL schema. Every element defined in the OSnL schema is an `OSnLNode` element that is of type `OSnLNode`. This leads to very efficient parsing.
  We take a very object-oriented approach. An `OSnLNode` is much like an abstract class in C++ or Java.

OSProtocols: Representation

The representation protocols are standards based on XML schemas. Several of the key representation standards include:

- **OSrL:** Optimization Services Result Language. This a standard for solver (server) to communicate back to the modeling language (client) the result of the optimization.
- **OSoL:** Optimization Services Option Language. This is a standard for communicating options to a solver, e.g. solve using dual simplex.

There is also **OSInstance** that is a standard for representing problem instances in-memory. Classes in OSInstance correspond to ComplexTypes in the corresponding XML Schema.

OS Protocols: Representation

The representation standards are all XML based. The information is in a text file with data and markup.

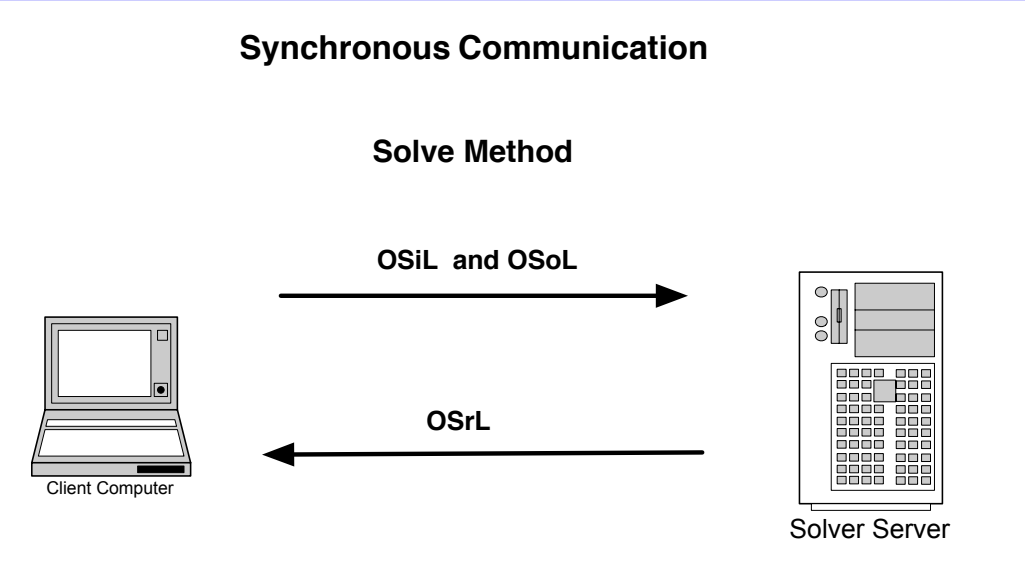
```
<variables>
     <values>
          <var idx="1" value="540.2"/>
          <var idx="2" value="241.9"/>
          <var idx="3" value="34"/>
     </values>
</variables>
```

OSProtocols: Communication

The key protocol is Optimization Service Hookup Language (OShL). A set of methods that control communication between a client and a server.

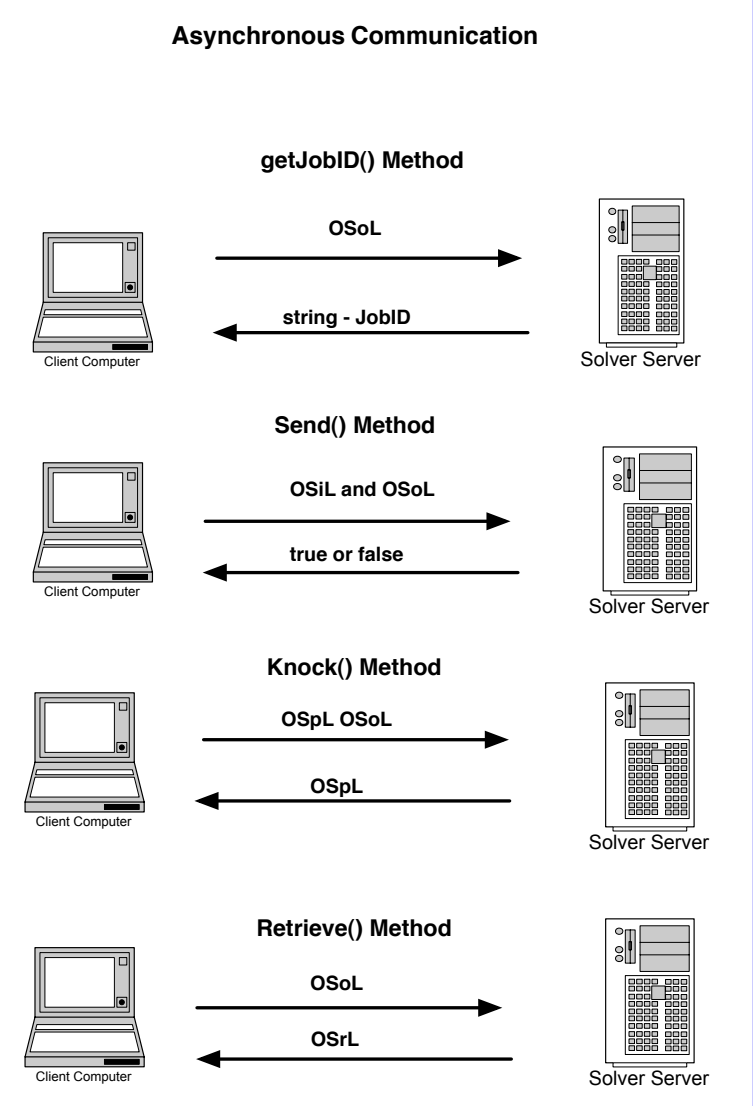
Synchronous Service:

- `solve(xsd__string osil, xsd_string osol)`



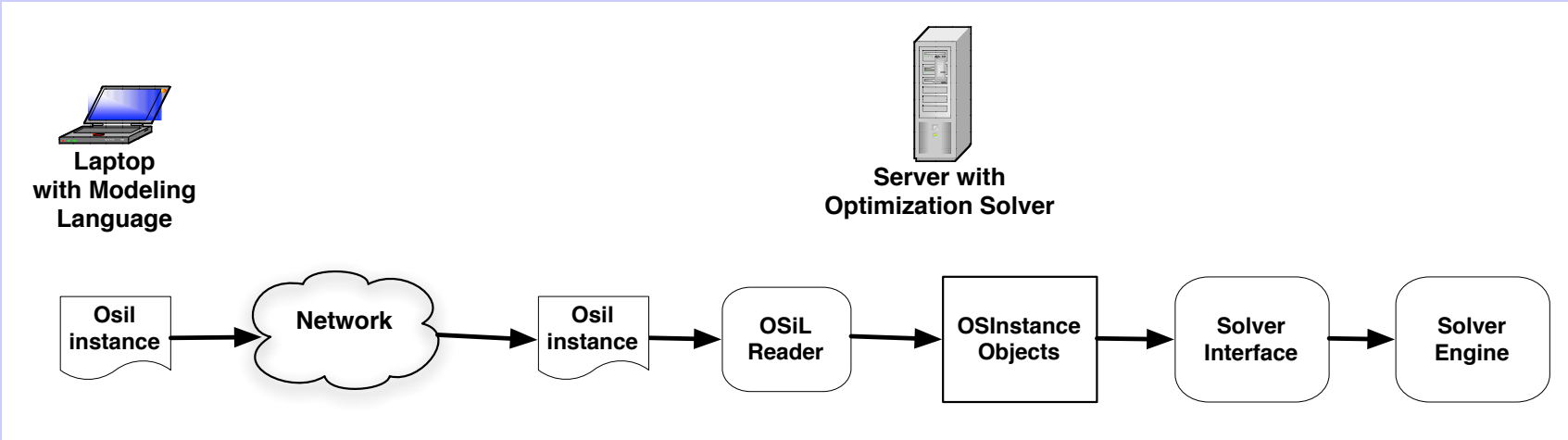
OSProtocols: Communication

Asynchronous Communication and Calls:



Solver Service – An OS Implementation

Optimization Services is a set of protocols. We now describe server and client services built on optimizations services.



Solver Service – An OS Implementation

Implementing a Solver Service

- On the solver end expose an optimization solver (or problem analyzer).
- This is most easily done by using an existing Web Server that supports Web Services.
  1. Apache + Tomcat
  2. Tomcat (Java or C++) – We have implemented both.
  3. JBoss
  4. IIS
  5. High end – Websphere, Weblogic, Oracle, Geronimo
- Programming language is irrelevant but Java dominates the XML world.

Solver Service – An OS Implementation

Implement the OS Communication standard:

- `solve(osil, osol)`:
  - Takes OSiL and OSoL and returns OSrL (string/file version)
  - Synchronous call, blocking request/response
- `getJobID(osol)`
  - Gets a unique job id generated by the solver service
  - Maintain session and state on a distributed system
- `send(osil, osol)`
  - Same signature as the solve function but returns a boolean
  - Asynchronous (server side), non-blocking call
- `knock(ospl, osol)`
  - Get process and job status information from the remote server
- `retrieve(osol)`
  - Retrieving result from anywhere anytime
- `kill(osol)`
  - kill remote optimization jobs
  - Critical in long running optimization jobs

Solver Service – An OS Implementation

The `solve(osil, osol)` method in detail for a CoinSolverService.

```
string OSSolverService::solve(string osil, string osol)
{
     OSServiceUtil serviceUtil;
     serviceUtil.m_solver = new CoinSolver();
     char* osrl = &serviceUtil.solve(osil, osol)[0];
     return osrl;
}
```

The `serviceUtil` object used above has a data member that is the implementation of an abstract `DefaultSolverClass`.
If you use the OS libraries you do not need to worry about `getJobID()`, `send()` `kill()`, `knock()`, and `retrieve()`. They are done for you.

Solver Service – An OS Implementation

OSSolver Library: If you use this library (available in both Java and C++) it is trivial to hook your solver into our API.

- The API provides a `DefaultSolver` class. It is an abstract class.
- The `DefaultSolver` class has the pure virtual function

```
virtual string solve(string osil, string osol) = 0;
```

- Define your own class that inherits from `DefaultSolver` and implements the `solve()` method for your solver.

Solver Service – An OS Implementation

Programming language becomes important at this point. What if, e.g. your solver is C++ and the Web Service is in Java. Two options.

- Option 1: Use JNI (Java Native Interface). Call the `solve()` method using JNI. A bit risky.
- Option 2: Implement a Java `solve()` method in the Java Web Service that use the Java Runtime class. Use this to launch a C++ executable. Pass the executable the OSiL and OSoL as files. Then have the executable write a file with the OSrL and pass this back in the SOAP envelope.

The OSSolver Library - COIN Solver Example

Here is how the CoinSolver class works.

```
class CoinSolver : public DefaultSolver{
public:
     string solve(string osil, string osol);
};
```

Now implement the CoinSolver solver.

```
string CoinSolver::solve(string osil, string osol)  {
     solverName = osol;
     OSiLReader* osilreader;
     OSInstance* theosinstance = 0;
     OsiSolverInterface* solver = 0;
```

⋮

The OSSolver Library - COIN Solver Example

Implementation of CoinSolver solve() method (continued).

```
    if(osol == "glpk")
        solver = new OsiGlpkSolverInterface();
    else
        solver = new OsiClpSolverInterface();
    osilreader->readOSiL( osil );
    theosinstance = osilreader->getOSInstance();
    if(!setOSInstance( theosinstance )) return 0;
    return optimize();
}
```

Important: the CoinSolver class must put the OSInstance object into the COIN data structures such as the CoinPackedMatrix

Client Service – An OS Implementation

The SOAP – first there is a header

```
POST /os/ossolver/LindoSolverService.jws HTTP/1.0
Host: gsbkip.chicagogsb.edu
Content-Type: text/xml; charset=UTF-8
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: "OSSolverService#solve"
Content-Length: 2335
```

The key line is the POST command. It tells the server which service to use. In this case it is LINDO.

Client Service

The SOAP – then the **envelop.** In this case we are implementing a **solve** over the network.

```
<SOAP-ENV:Envelope>
     <SOAP-ENV:Body>
         <solve>
              <osil xsi:type="xsd:string">
                    The OSiL string goes here
              </osil>
              <osol xsi:type="xsd:string">
                    The OSoL string goes here
             </osol>
         </solve>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Client Service

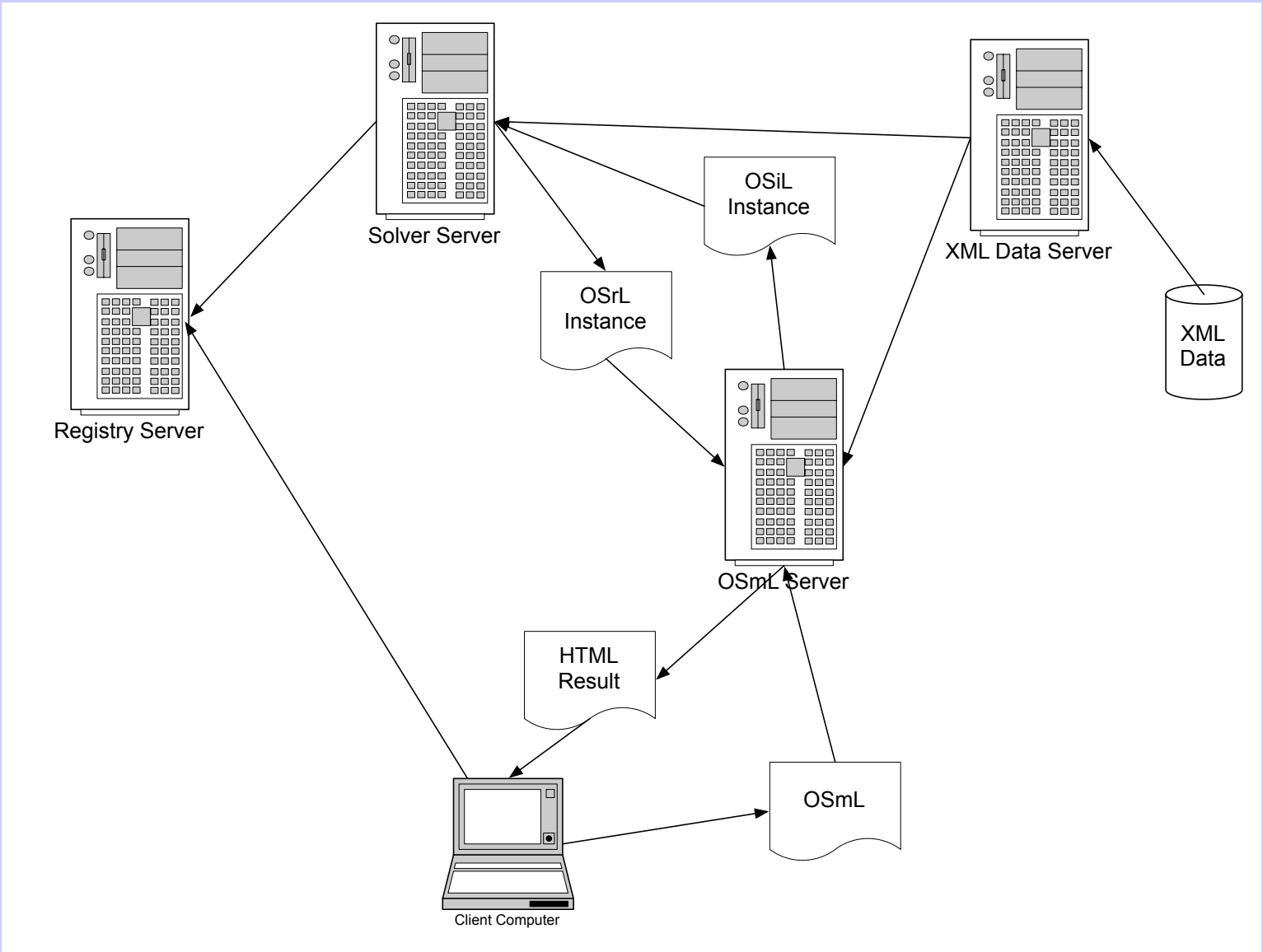
WSDL – Web Services Discovery Language.

You can use products such as Apache Axis and Visual Studio .NET to generate code from the WSDL.

For example:

`http://gsbkip.chicagogsb.edu/os/osolver/COINSolverService.jws?`

Summary: An Example of Optimization Services



Summary

Modeling languages that can generate OSiL:

- AMPL (linear OSiL – use nl2osil.exe)
- OSmL (native linear and nonlinear)
- POAMS (native linear OSiL???)

Solvers:

- CLP - through COIN OSI
- FORTMP - LPFML
- GLPK – through COIN OSI
- IMPACT - native support
- KNITRO - using function callbacks
- LINDO – using instruction list format