Optimization Services: A Framework for Distributed Optimization

Robert Fourer and Jun Ma
Northwestern University

Kipp Martin
University of Chicago

June 2, 2006

Outline

Optimization Services (OS)

Service Oriented Architectures and Web Services

Optimization As a Web Service

OS Protocols

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

Solver Service – An OS Implementation

COIN-OR

Client Service – An OS Implementation

Summary and Future Work

Key Themes

- **Main Idea:** It is necessary for OR people to cater to the IT community and use their tools, not the other way around!
- Witness the success of Excel Solver – the OR community got that one right.
- Key IT Technologies/Trends
    1. Extensible Markup Language (XML) for Data
    2. Web Services
    3. Service Oriented Architectures
    4. Software as service

Corollary 1: The OR community **must** use these technologies in order to integrate optimization into a modern IT infrastructure.

Optimization Services (OS)

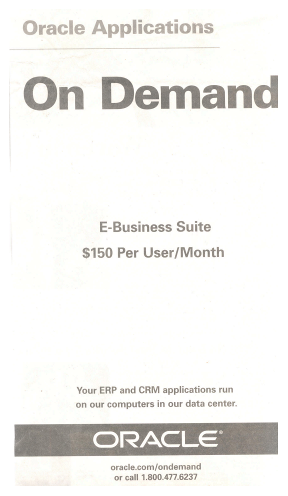
Software as a service! In industry, CRM (customer relationship software), tax preparation, Microsoft Office Live, etc. are all becoming services. All of the major players in software are promising software as a service. There clearly is a trend away from the fat client loaded with lots of heavyweight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

Corollary 2: Optimization should available as a software service. It should be easy to solve optimization problems of any type (linear, integer, nonlinear, stochastic, etc), at any time, if you are hooked up to the network.

Optimization Services is our attempt to make optimization a service.

Optimization Services (OS)

If enterprise software is offered as a service, but optimization is not, how can we possibly hope to have optimization integrated into these products?



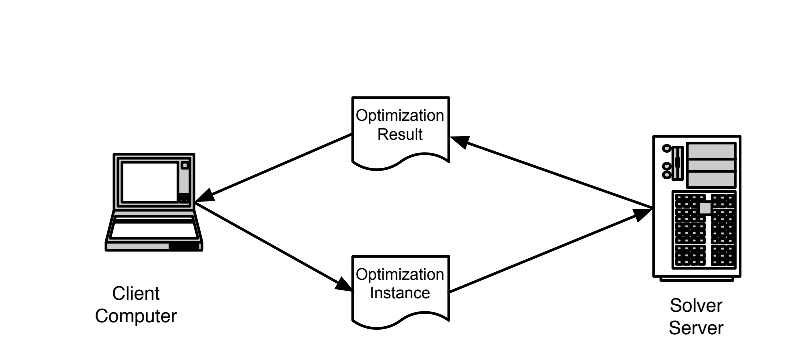
Optimization Services (OS)

"I commend the work you've done on the XML representation of linear programming problems. Im with a Business Intelligence start-up in California that would like to incorporate LP into our solution and were evaluating a number of options including the LPFML specification."

An email received May 31, 2006 from the CEO of a Business Intelligence start-up.

Optimization Services (OS)

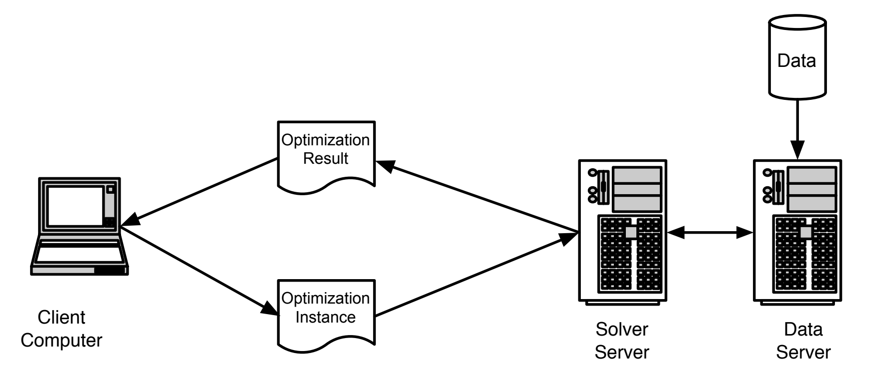
A simple scenario:



Take advantage of a faster machine (the server), code not on the client, a better license deal, open source software, etc. Maintaining code on a single machine is just easier.

Optimization Services (OS)

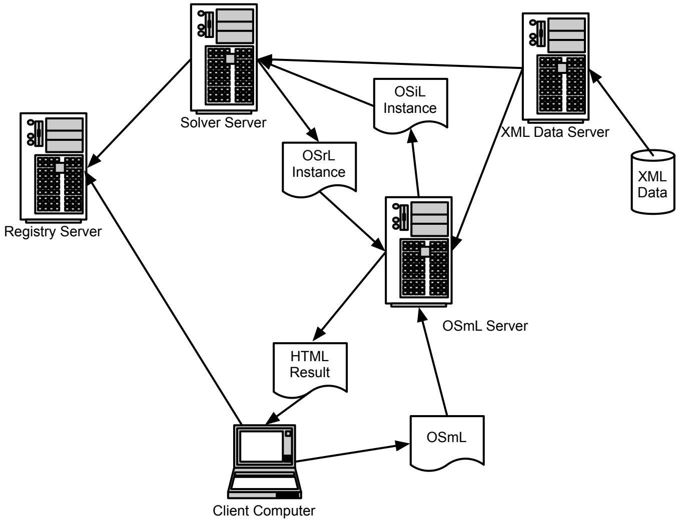
A more complicated scenario:



In a realistic modeling environment we cannot expect the data to be on the client machine. We may also want a feed to real time data to make sure the problem instance is current.

Optimization Services (OS)

Optimization Services on **Steroids!!!**



Optimization Services (OS)

Optimization Services:

- *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 should be as easy as hooking up to the network.*

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.

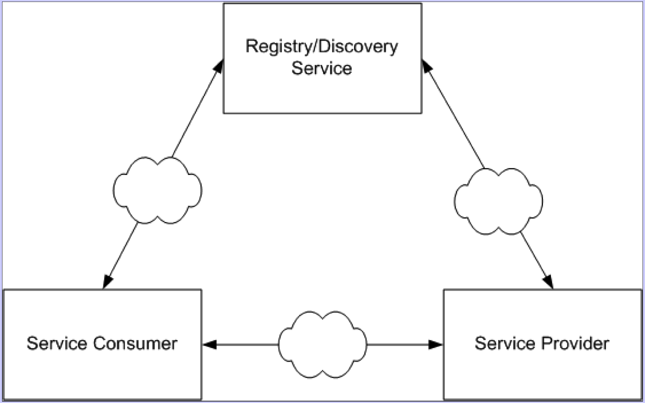
Optimization Services

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:

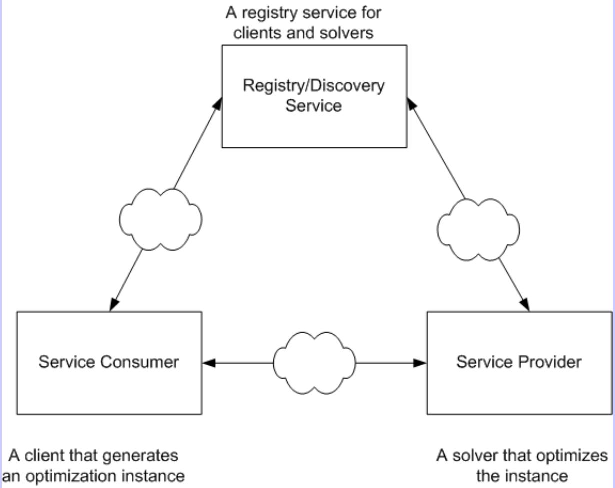
- Uses open standards, e.g. HTTP, XML, SOAP
- Can be used to develop rich clients
- Can be used by components people not necessary

Web Services makes use of three major protocols:

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

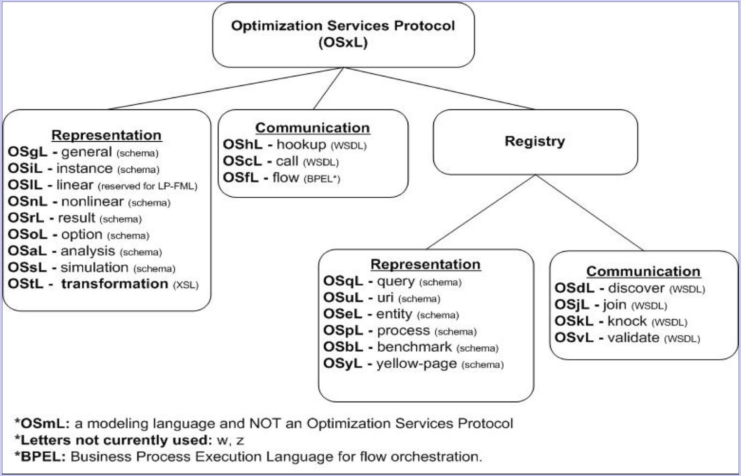
Optimization As a Web Service

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



OS Protocols

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



OS Protocols: Representation

- In a distributed setting the model may be generated on one machine and the model optimized on another machine
- The solver wants an instance as opposed to a model
- **Important Distinction:** model versus instance

Model versus Instance (See Fourer 83)

Model	**Instance**
Symbolic	Explicit
General	Specific
Concise	Verbose
Understandable	Convenient

OS Protocols: Representation

A simple production scheduling **model** in AMPL

```
set PROD;   # products
set DEP;    # processing departments

param hours {DEP};
param rate {DEP, PROD};
param profit {PROD};
var Make {PROD} >= 0;

maximize TotalProfit:
sum {j in PROD} profit[j] * Make[j];

subject to HoursAvailable {i in DEP}:
sum {j in PROD} rate[i,j] * Make[j] <= hours[i];
```

OS Protocols: Representation

Raw Data for a model

```
param: PROD: profit :=
             std    10
             del     9 ;

param: DEP:            hours :=
          cutanddye       630
          sewing          600
          finishing       708
          inspectandpack 135 ;

param: rate:                          std   del  :=
              cutanddye               0.7   1.0
              sewing                  0.5   0.8333
              finishing               1.0   0.6667
              inspectandpack          0.1   0.25   ;
```

OS Protocols: Representation

Instance = Model + Data

```
maximize TotalProfit:
10*Make[std] + 9*Make[del];

subject to HoursAvailable[cutanddye]:
0.7*Make[std] + Make[del] <= 630;

subject to HoursAvailable[sewing]:
0.5*Make[std] + 0.8333*Make[del] <= 600;

subject to HoursAvailable[finishing]:
Make[std] + 0.6667*Make[del] <= 708;

subject to HoursAvailable[inspectandpack]:
0.1*Make[std] + 0.25*Make[del] <= 135;
```

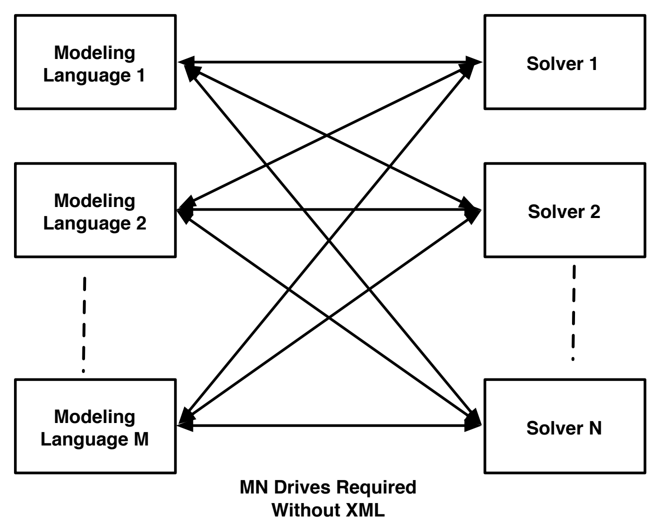
OS Protocols: Representation

A Proliferation of Modeling Languages and of Solvers

Modeling languages	**Solvers**
AIMMS	CLP
AMPL	CPLEX
GAMS	GLPK
LINGO	LINDO
Mosel	MINOS
MPL	MOSEK
OPL	Xpress-MP

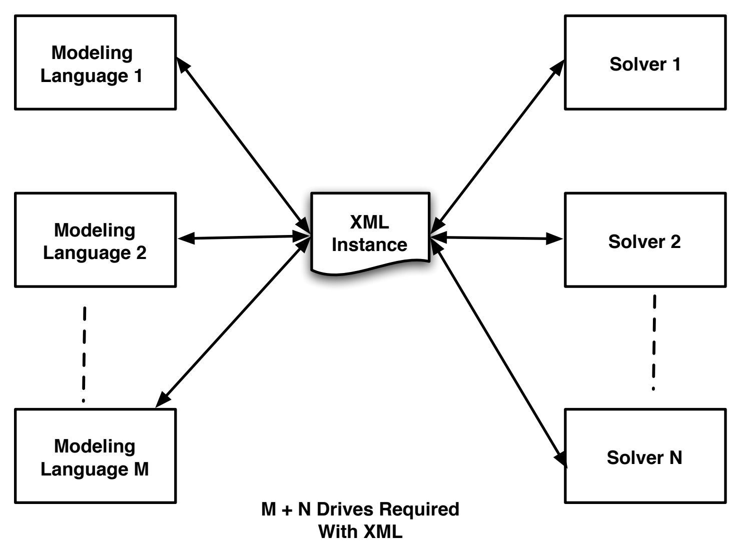
OS Protocols: Representation

Consequence: a lot of drivers are need for every modeling language to talk to every solver



OS Protocols: Representation

An **instance representation language** is required!



OS Protocols: OSiL

The protocol we developed for representing a broad variety of optimization problem instances is OSiL (Optimization Services instance Language).

The OSiL is defined using XML (Extensible Markup Language). The decision to use XML goes back to the initial theme of the talk.

- XML is rapidly becoming an accepted format for transferring/storing data. This is where the data is! Think Willie Sutton and Sam Savage.
- People in IT use XML. OR people should use IT tools, rather than having IT people use OR tools.

OS Protocols: OSiL

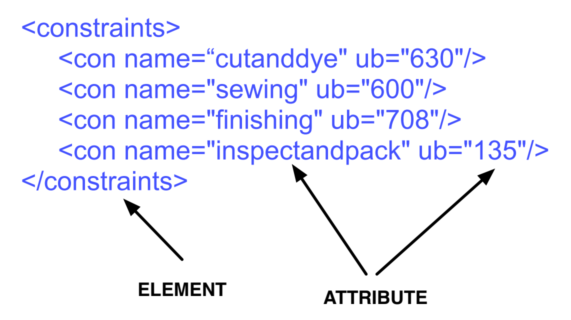
XML – A file that contains both **data** and **markup.** A very simple idea, yet very powerful. For example, here is how AMPL would store constraint information for a problem instance

```
n0
r
1 630
1 600
1 708
1 135
b
2 0
2 0
k1
```

The file is all data – very hard to parse! Contrast this with XML.

OS Protocols: OSiL

The XML file contains both data and Markup (Elements (tags) and Attributes).



Note: HTML (actually XHTML) is an example of an XML vocabulary. HTML is about formatting. For example in HTML you might write.

```
<p><b> See Spot Run</b></p>
```

OS Protocols: OSiL

$$
\begin{aligned}
\text{Minimize} \quad & (1 - x_0)^2 + 100(x_1 - x_0^2)^2 + 9x_1 \\
\text{Subject to} \quad & x_0 + 10x_0^2 + 11x_1^2 + 3x_0x_1 \leq 25 \\
& \ln(x_0x_1) + 7x_0 + 5x_1 \geq 10 \\
& x_0, x_1 \geq 0
\end{aligned}
$$

OS Protocols: OSiL

The variables: $x_0, x_1 \geq 0$

```
<variables number="2">
     <var lb="0" name="x0" type="C"/>
     <var lb="0" name="x1" type="C"/>
</variables>
```

The objective function: $\text{minimize} \quad 9x_1$

```
<objectives number="1">
     <obj maxOrMin="min" name="minCost">]
          <coef idx="1">9</coef>
     </obj>
</objectives>
```

OS Protocols: OSiL

The linear terms are stored using a sparse storage scheme

$$
\begin{aligned}
x_0 + 10x_0^2 + 11x_1^2 + 3x_0x_1 &\leq 25 \\
7x_0 + 5x_1 + \ln(x_0x_1) + &\geq 10
\end{aligned}
$$

```
<linearConstraintCoefficients>
     <start>
          <el>0</el><el>2</el><el>3</el>
     </start>
     <rowIdx>
          <el>0</el><el>1</el><el>1</el>
     </rowIdx>
     <value>
          <el>1.0</el><el>7.0</el><el>5.0</el>
     </value>
</linearConstraintCoefficients>
```

OS Protocols: OSiL

Representing quadratic and general nonlinear terms

$$\begin{aligned} x_0 + 10x_0^2 + 11x_1^2 + 3x_0x_1 &\leq 25 \\ 7x_0 + 5x_1 + \ln(x_0x_1) + &\geq 10 \end{aligned}$$

```
<quadraticCoefficients numberOfQuadraticTerms="3">
     <qTerm idx="0" idxOne="0" idxTwo="0" coef="10"/>
     <qTerm idx="0" idxOne="1" idxTwo="1" coef="11"/>
     <qTerm idx="0" idxOne="0" idxTwo="1" coef="3"/>
</quadraticCoefficients>

<nl idx="1">
     <ln>
          <times>
               <variable coef="1.0" idx="0"/>
               <variable coef="1.0" idx="1"/>
          </times>
     </ln>
</nl>
```

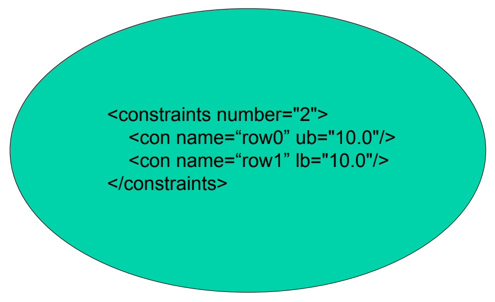
OS Protocols: OSiL

Key idea a **schema**. How do we know how to write proper OSiL? Similar to the concept of a class in object orient programming. Critical for parsing!

Schema $\Longleftrightarrow$ **Class**

XML File $\Longleftrightarrow$ **Object**

We need a schema to define the OSiL instance language.

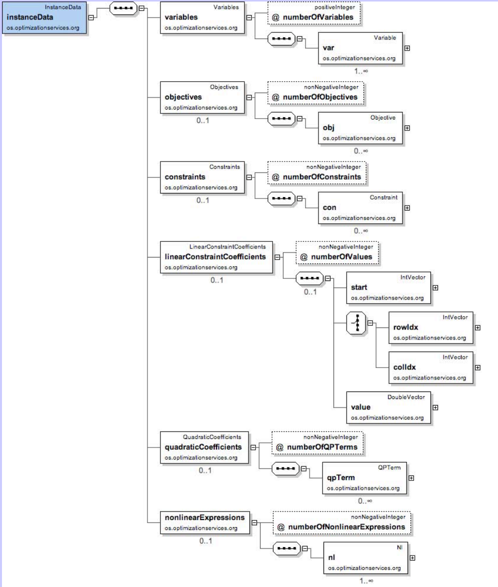


OS Protocols: OSiL

Schema a Constraints and Con Class

```xml
<xs:complexType name="constraints">
        <xs:sequence>
                <xs:element name="con" type="con" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="number" type="xs:nonNegativeInteger" use="required"/>
</xs:complexType>
<xs:complexType name="con">
        <xs:attribute name="name" type="xs:string" use="optional"/>
        <xs:attribute name="lb" type="xs:double" use="optional" default="-INF"/>
        <xs:attribute name="ub" type="xs:double" use="optional" default="INF"/>
        <xs:attribute name="mult" type="xs:positiveInteger" use="optional" default="1"/>
</xs:complexType>
```

The OSiL Schema



The OSiL Schema

The schema is used to **validate** the XML document. Think of validation as an error check.

The schema defines an XML vocabulary, language, or dialect. Examples include:

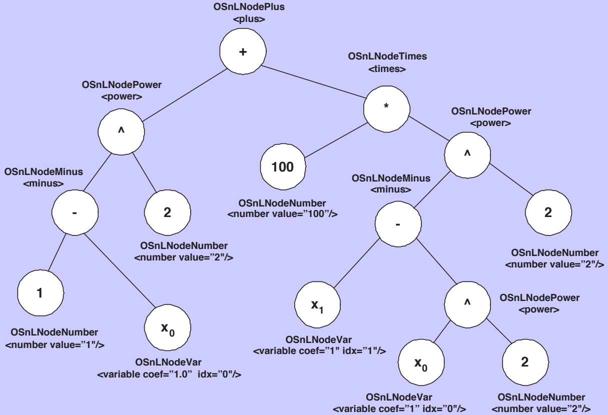
- XHTML – the markup language for Web documents
- FpML– Financial products Markup Language
- WordProcessingML and SpreadsheetML for Microsoft Office
- XBRL– eXtensible Business Reporting Language
- MathML– a format for representing math on Web pages
- AnatML– Anatomical Markup Language
- RSS – Really Simple Syndication for news feeds

OSiL – the markup language for optimization instances

The OSiL Schema

$$(1 - x_0)^2 + 100(x_1 - x_0^2)^2$$

How do we validate this? Designing the schema is a huge problem!



The OSiL Schema

Design Goal: *represent a comprehensive collection of optimization problems while keeping parsing relatively simple.* Not easy!!!

- For purposes of validation, any schema needs an explicit description of the children allowed in a `<operator>` element
- It is clearly inefficient to list every possible nonlinear operator or nonlinear function allowed as a child element. If there are n allowable nonlinear elements (functions and operators), listing every potential child element, of every potential nonlinear element, leads to $O(n^2)$ possible combinations.
- This is also a problem when doing function and gradient evaluations, etc. a real PAIN with numerous operators and operands.
- We avoid this by having EVERY nonlinear node an OSnLNode instance.

The OSiL Schema

Solution: *Use objected oriented features of the XML Schema standard.*

```
<xs:complexType name="OSnLNode" mixed="false"/>
<xs:element name="OSnLNode" type="OSnLNode"
        abstract="true">
```

The multiplication operator

Extend OSnLNode

```
<xs:complexType name="OSnLNodePlus">
   <xs:complexContent>
      <xs:extension base="OSnLNode">
         <xs:sequence minOccurs="2" maxOccurs="2">
            <xs:element ref="OSnLNode"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The OSiL Schema

- The code for implementing this is written in C++.
- The C++ code "mimics" the XML schema
- In C++ there is an abstract class **OSnLNode** with pure virtual functions for function and gradient calculation.
- There are operator classes such as **OSnLNodePlus** that inherit from **OSnLNode** and *do the right thing* using polymorphism.

```
m_mChildren = new OSnLNode*[2];
double OSnLNodePlus::calculateFunction(double *x){
   m_dFunctionValue = m_mChildren[0]->calculateFunction(x)
   + m_mChildren[1]->calculateFunction(x);
   return m_dFunctionValue;
}// end OSnLNodePlus::calculate
```

OS Protocols: Representation

Two other 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.

OS Protocols: OSrL

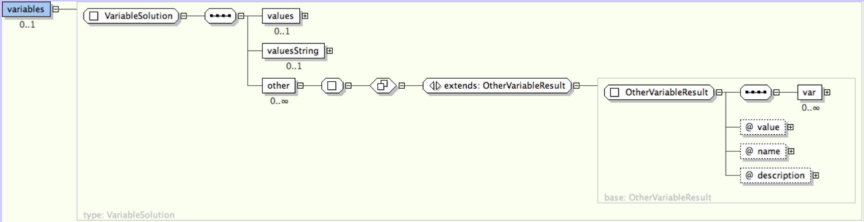
Here is an example of OSrL (Optimization Services result Language)

```
<variables>
     <values>
          <var idx="0">539.984</var>
          <var idx="1">252.011</var>
     </values>
</variables>
<objectives>
     <values>
      <obj idx="-1">7667.94</obj>
     </values>
</objectives>
```

The fact that the result is in XML has important implications. It is now easy to write XSLT (Extensible Stylesheet Language Transformation) stylesheets to transform the result into human readable HTML.

OS Protocols: OSrL

The use of the <other> element.



OS Protocols: OSoL

An example of Optimization Services result Language

```
<?xml version="1.0" encoding="UTF-8"?>
<osol >
<general>
     <instanceLocation locationType="http">
          http://gsbkip.chicagogsb.edu/parincLinear.osil
     </instanceLocation>
     <contact transportType="smtp">
          kipp.martin@chicagogsb.edu
     </contact>
</general>
</osol>
```

Two important features:

- the option to have result notifications sent via email (could also ftp)
- the option to specify a problem instance on a remote machine for solution

OS Protocols: Representation

Summary: The case for XML in EVERY protocol!

- Validation against a schema provides for error checking
- Validation against a schema promotes stability of a standard
- The schema can restrict data values to appropriate types, e.g. row names to string, indices to integer, coefficients to double
- The schema can define keys to insure, for example, no row or column name is used more than once
- The schema can be extended to include new constraint types or solver directives
- There is a lot of open source software to make parsing easy

OS Protocols: Representation

Summary: The case for XML in an optimization system.

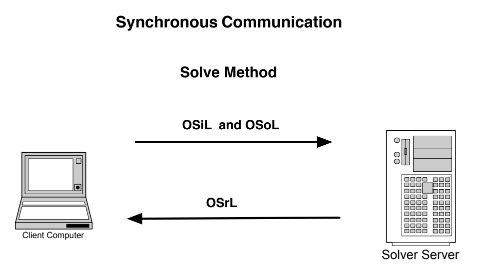
- When instances are stored in XML format, optimization technology solutions are more readily integrated into broader IT infrastructures
- XML is used for Web Services important for distributed computing
- The XML format lends itself well to compression
- The XML format can be combined with other technologies, e.g. XSLT to present results in human readable formats
- Encryption standards are emerging for XML possibly important in a commercial setting

OS Protocols: Communication

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

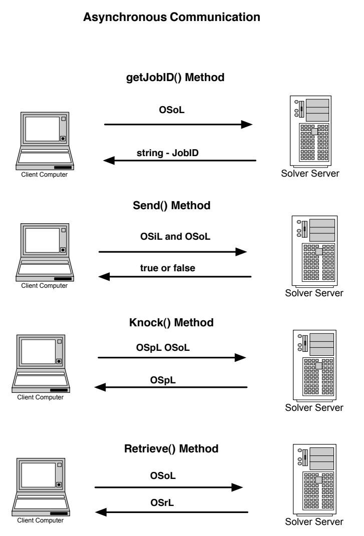
Synchronous Service:

- `solve(xsd__string osil, xsd_string osol)`



OS Protocols: Communication

Asynchronous Communication and Calls:



OS Protocols: Communication

Summary of Communication Protocols:

- `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

OS Protocols: Registry

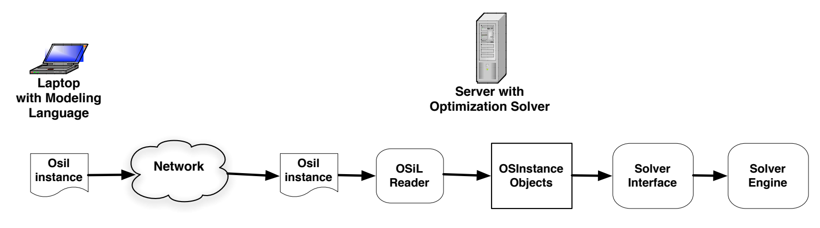
Two parts:

1. A client may not have the address of a solver. This problem is resolved by contacting the registry for a solver address.

2. A solver may wish to register its service with the registry.

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

We have a set of open-source Java libraries that implement our communication protocols of **solve(), send(), knock(), kill(), retrieve(), getJobID()**.

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. This is what we do.

Solver Service – An OS Implementation

In our implementation (again all open source) there is an **OSSolverService** executable in C++ (and to be made available on Windows, Linux, and Mac platforms).

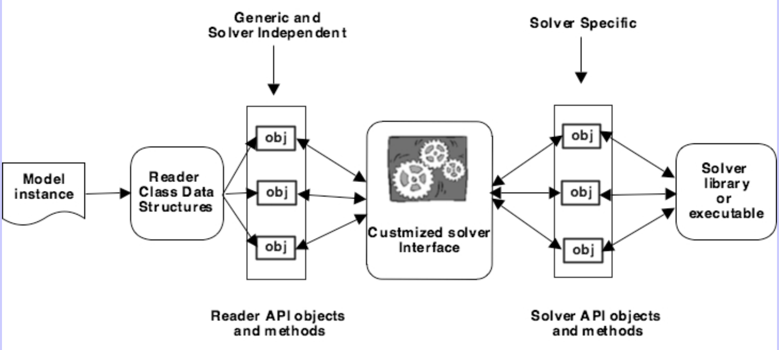
Here is what OSSolverService does:

- Read the OSoL file
- Determine the appropriate solver and instantiate a solver object in the appropriate solver class
- Pass the appropriate solver object the instance in OSiL format

The **OSSolverService** is linked with the necessary solver libraries.

Solver Service – An OS Implementation

Here is a generic implementation of a solver object.



Solver Service: OSInstance API

When the problem instance is read into memory an **OSInstance** object is created that provides an API (Application Program Interface) to the problem data for the solver.

Schema complexType | **In-memory class**

```
<xs:complexType name="Variables">  <---------------------------------------------->  class Variables{
                                                                                       public;
  <xs:sequence>                                                                          Variables();
    <xs:element name="var" type="Variable" maxOccurs="unbounded"/>  <------------->      Variable *var;
  </xs:sequence>
  <xs:attribute name="number" type="xs:positiveInteger"  use="required"/>  <------>      int number;
</xs:complexType>                                                                      }; // class Variables


<xs:complexType name="Variable">  <----------------------------------------------->  class Variable{
                                                                                       public;
                                                                                         Variable();
  <xs:attribute name="name" type="xs:string" use="optional"/>  <------------------>      string name;
  <xs:attribute name="init" type="xs:double" use="optional"/>  <------------------>      double init;
  <xs:attribute name="initString" type="xs:string" use="optional"/>  <------------>      string initString;
  <xs:attribute name="type" use="optional" default="C">  <------------------------>      char type;
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="C"/>
        <xs:enumeration value="B"/>
        <xs:enumeration value="I"/>
        <xs:enumeration value="S"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="lb" type="xs:double" use="optional" default="0"/>  <-------->      double lb;
  <xs:attribute name="ub" type="xs:double" use="optional" default="INF"/>  <------>      double ub;
</xs:complexType>                                                                      }; // class Variable
```

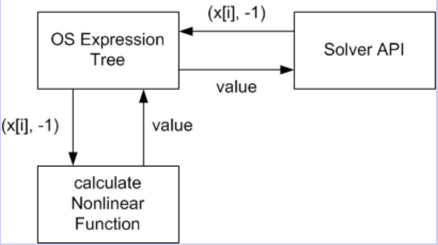
OSiL elements | **In-memory objects**

```
<variables number="2">                      OSInstance osintance;
   <var lb="0" name="x0" type="C"/>         osinstance.instanceData.variables.number=2;
   <var lb="0" name="x1" type="C"/>         osinstance.instanceData.variables.var=new Var[2];
</variables>                                osinstance.instanceData.variables.var[0].lb=0;
                                            osinstance.instanceData.variables.var[0].name=x0;
                                            osinstance.instanceData.variables.var[0].type=C;
                                            osinstance.instanceData.variables.var[1].lb=0;
                                            osinstance.instanceData.variables.var[1].name=x1;
                                            osinstance.instanceData.variables.var[1].type=C;
```

Solver Service – OSInstance API

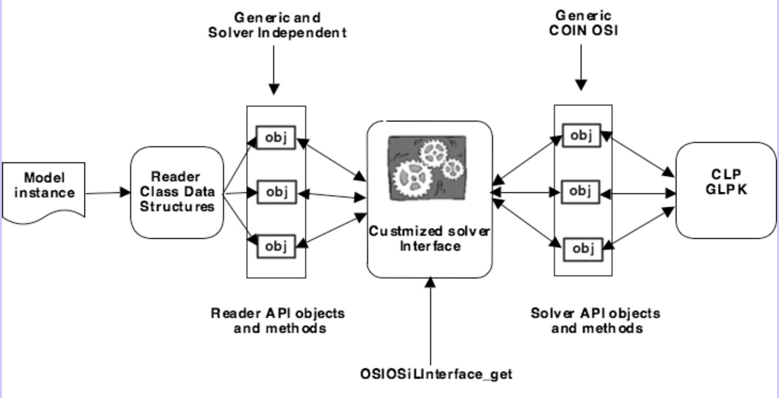
Some OSInstance API features are:

- get instruction lists in postfix or prefix
- get a text version of the model in infix
- get function and gradient evaluations
- get information about constraints, variables, objective function, the A matrix, etc.
- get the root node of the OSExpression tree



Solver Service – An OS Implementation

Here is an implementation for solvers that can be linked to the COIN OSI.



COIN-OR

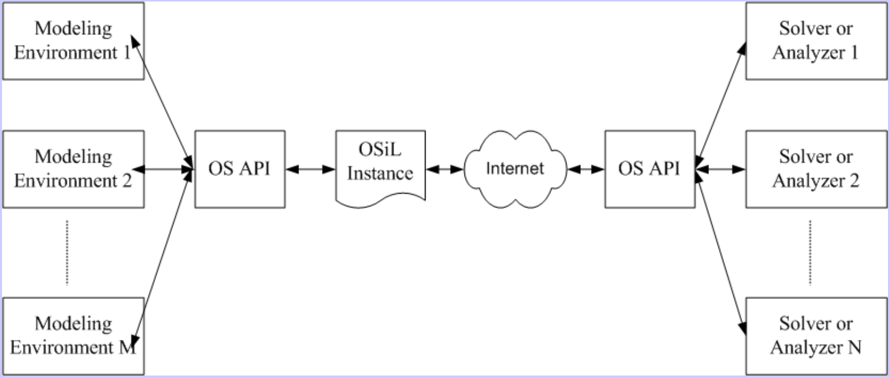
COIN-OR COmputational INfrastructure for OR

- The open-source movement has come to OR
- High-quality solvers are available under the CPL (common public license).
- See for example CLP (linear), IPOPT (nonlinear), CBC (integer)
- You can download either the source or binaries for the major platforms
- Have an idea for a new IP cut? Piggyback off the code already there.
- Why pay for what you can get for free?
- There is a conference at DIMACS July 17-20 with a lot of tutorials.

See www.coin-or.org.

Client Service – An OS Implementation

Here is what happens on the client end.



Client Service – An OS Implementation

The SOAP – first there is a header

```
POST /lindo/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 **envelope** 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

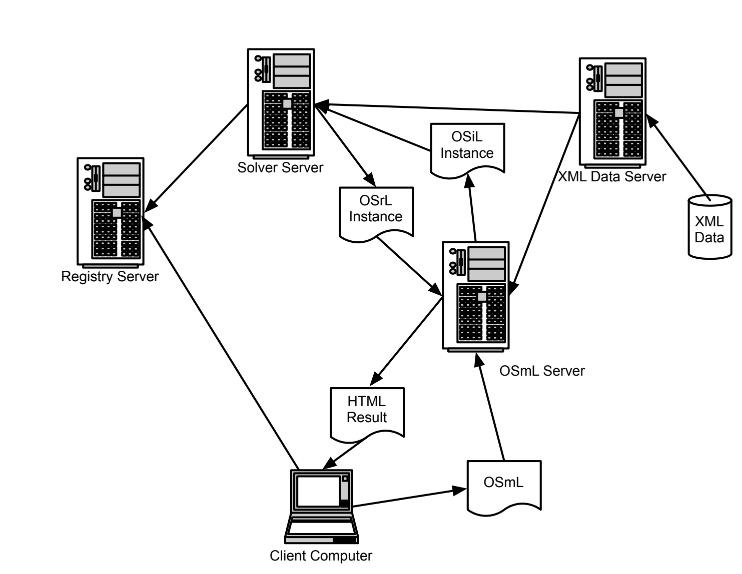
WSDL – Web Services Discovery Language.

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

For example:

`http://128.135.130.17:8080/lindo/LindoSolverService.jws?`

Summary: An Example of Optimization Services



Summary and Future Work

Modeling languages that can generate OSiL:

- AMPL (linear OSiL – use nl2osil class – nonlinear on the way)
- OSmL (native linear and nonlinear)
- POAMS (native linear OSiL???)

Solvers:

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

Future Work: To Do List

1. Finish libraries and donate to COIN-OR. All of this work will be available under the CPL.
2. Work on extensions to OSiL
   - constraint programming
   - cone programming
   - disjunctive and piecewise linear
   - user defined functions
   - real time data through XPath
   - stochastic programming
3. A complete remake of the client GUI.
4. Hook the system up to Excel so you can formulate the model in Excel but call any solver remotely. Use the Web Service References Tool in Microsoft Office Visual Basic Editor.

Future Work: To Do List

5. **OSmL:** Optimization Services modeling Language

   A. It should be able to act as an agent and send OSiL files to a server with a solver that implements Optimization Services

   B. It should be a true algebraic modeling language

      1. take a general infix notation
      2. support sets and subscripts
      3. have looping capability
      4. support logical conditions
      5. allow for user-defined functions
      6. allow for sparse sets, union, intersection, etc.

   C. Store model instances internally as an OSInstance object

   D. Access XML data using XPath

The OSmL Philosophy: All X all the time!

Key Premise: OSmL is based on **XQuery**. Think of XQuery as a much more powerful SQL applied to XML data rather than relational data.

SQL:

- SELECT
- FROM
- WHERE

XQuery (FLWOR flower):

- For
- Where
- Let
- Order by
- Return

The OSmL Philosophy: All X all the time!

Key Premise: XQuery (an extension of XPath) is a very powerful modeling language for mathematical optimization. It is (See Fourer 1983):

- symbolic
- general
- concise
- understandable

We can build a modeling language using existing W3C standards!

The OSmL Philosophy