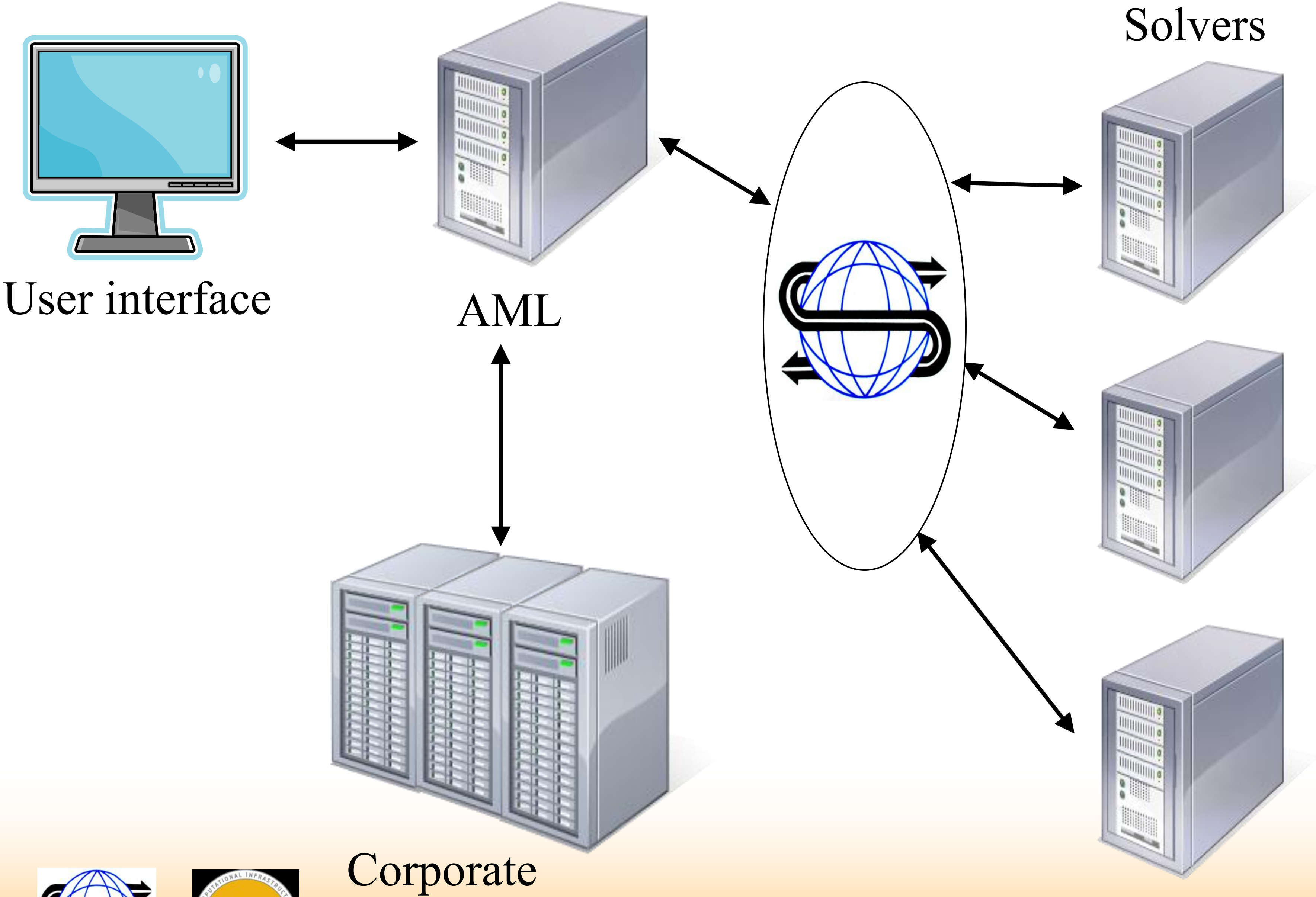
Optimization Services: Communicating Solver Options and Solver Results

H.I. Gassmann, Dalhousie University
J. Ma, Breakthrough Technologies
R.K. Martin, The University of Chicago

INFORMS, Charlotte NC, November 2011

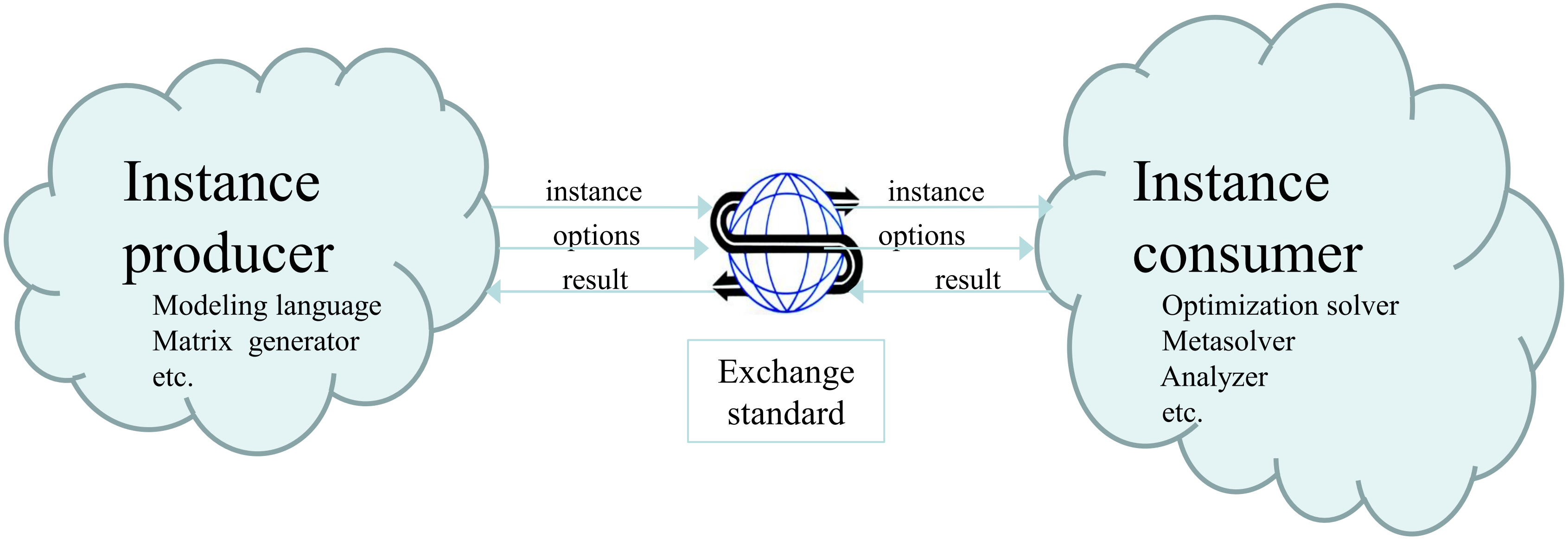
Outline

- Distributed computing and OR
- Solver options
- OSoL – OS option language
- Solver results
- OSrL – OS result language
- Availability



© 2011 H.I. Gassmann

Another way to look at it…



Instance producer

Modeling language
Matrix generator
etc.

instance

options

result

Exchange standard

instance

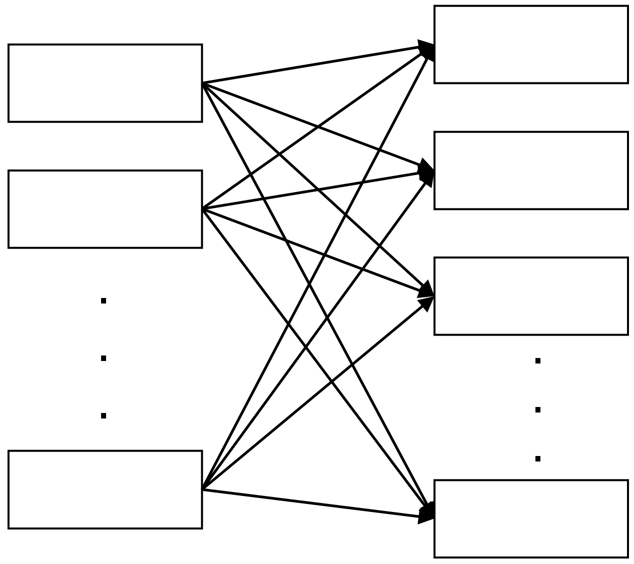
options

result

Instance consumer

Optimization solver
Metasolver
Analyzer
etc.

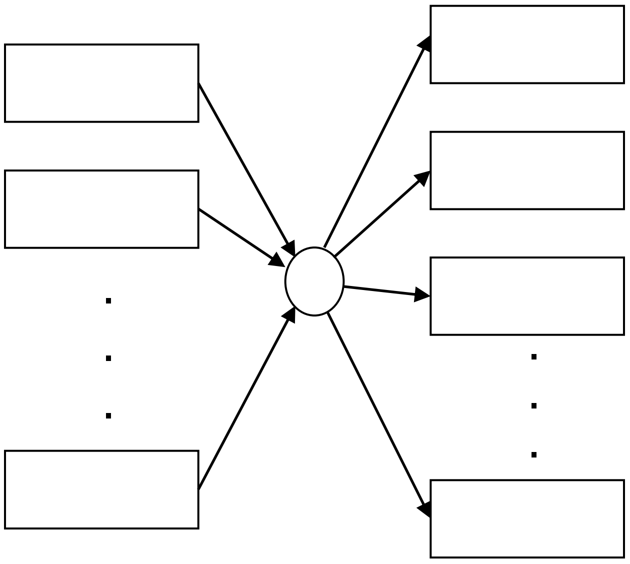
Why a standard interface?



Modelling systems

Solvers

*n***m* hook-ups



Modelling systems

Solvers

n+*m* hook-ups

Why a standard interface?

- Numerous modeling languages each with their own format for storing the underlying model
- Numerous solvers each with their own API
- Numerous operating system, hardware, and programming language combinations
- No standard for representing problem instances, especially nonlinear optimization instances
- No real standard for registry and discovery services

Separation of functionality

- Need to represent
  - Instance
  - Option
  - Result
  - Modifications

Instance vs. options

- Instance describes what is to be solved
  - Variables, objectives, relationships
- Options explain how to solve it
  - Algorithm tuning
    - e.g., tolerances, pricing and branching rules
  - Job performance
    - e.g., iteration limits, CPU limits
  - System requirements
  - Other, e.g., control of output levels
- **BUT**: branching weights, starting points
- One instance may be input into many solvers
- Solver options usually cannot be shared

Solver option characteristics

- Different classes of options
- Many options shared among solvers
- Some options unique to one solver
- Syntax and meaning may vary

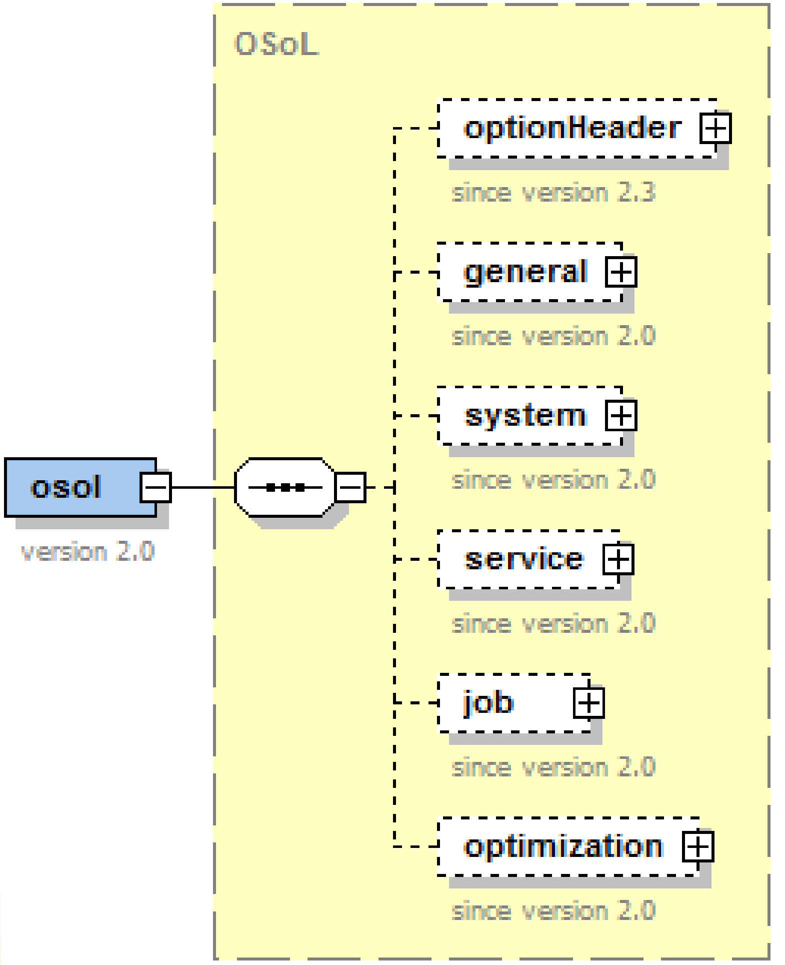
OSoL – OS option Language

- XML-based
- Common syntax
- Solver-specific semantics
- Standard representation for common options
- Flexibility to allow extensions
- Solver driver translates options into form understandable by the solver
- In-memory representation: `OSOption`
- API: `get()`, `set()`, `add()` methods

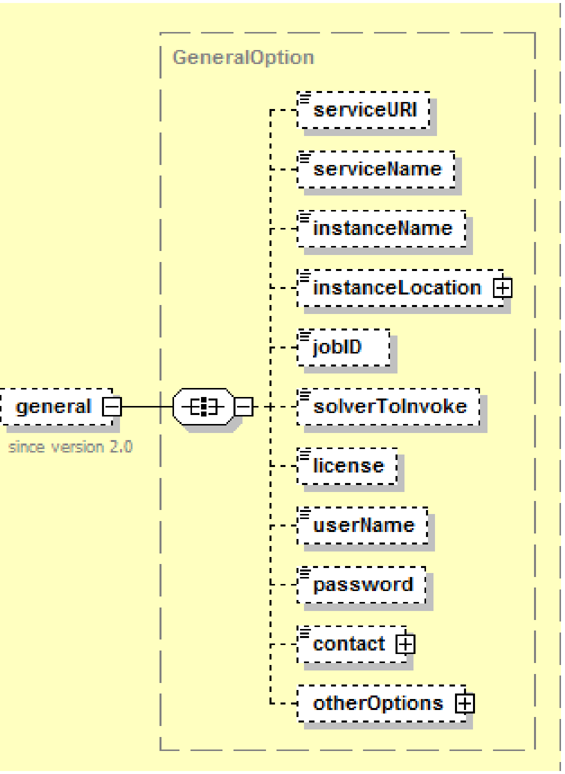
Why XML?

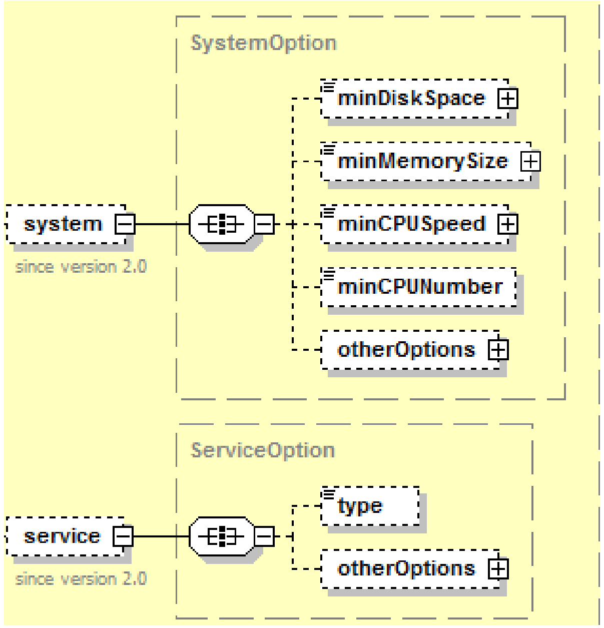
- Existing parsers to check syntax
- Easy to check, verify and impose compliance with standard
- Easy to generate automatically
- Automatic attribute checking (e.g., nonnegativity)
- Easy and natural transcription into in-memory objects
- Encryption standards being developed
- Easy integration into broader IT infrastructure

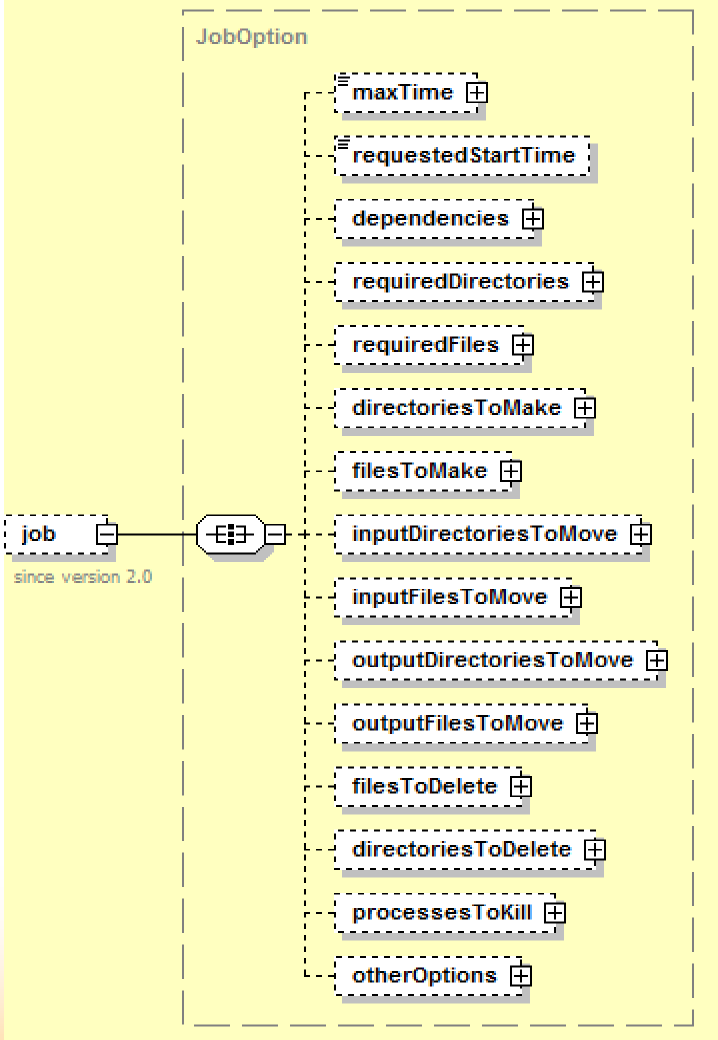
OSoL schema


OSoL
osol
version 2.0
optionHeader
since version 2.3
general
since version 2.0
system
since version 2.0
service
since version 2.0
job
since version 2.0
optimization
since version 2.0

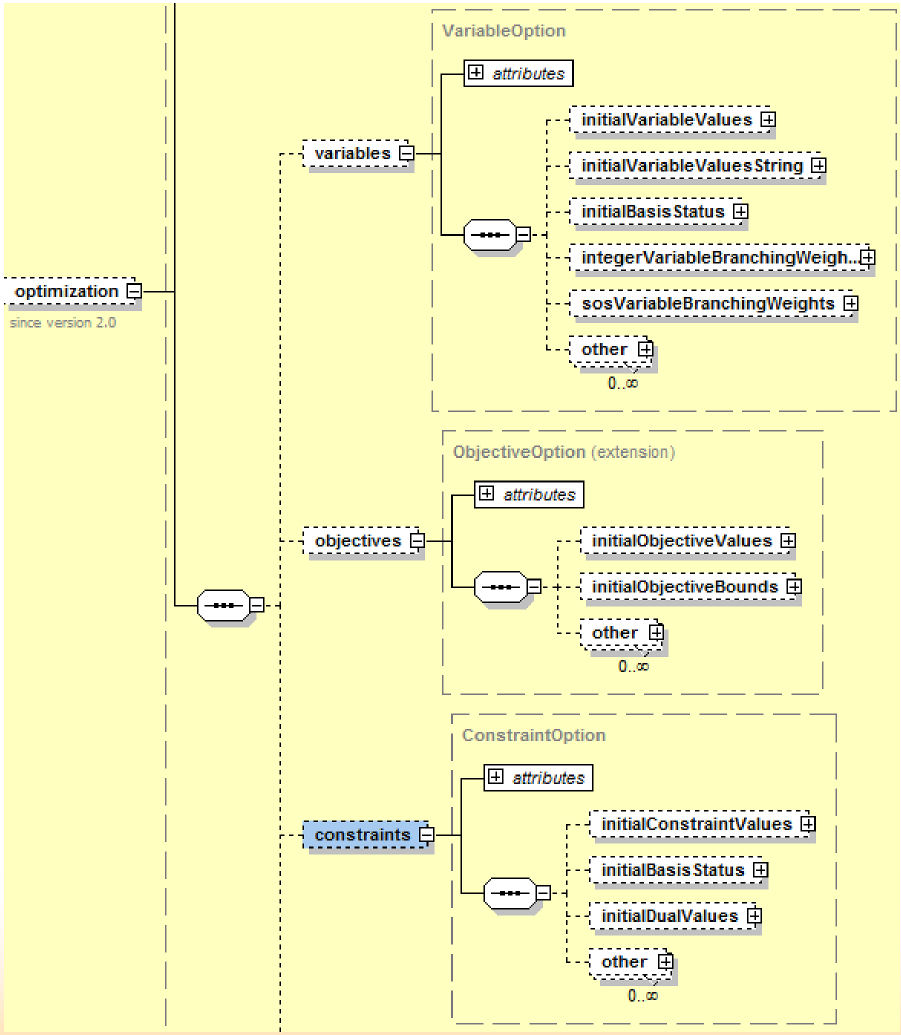
OSoL schema elements





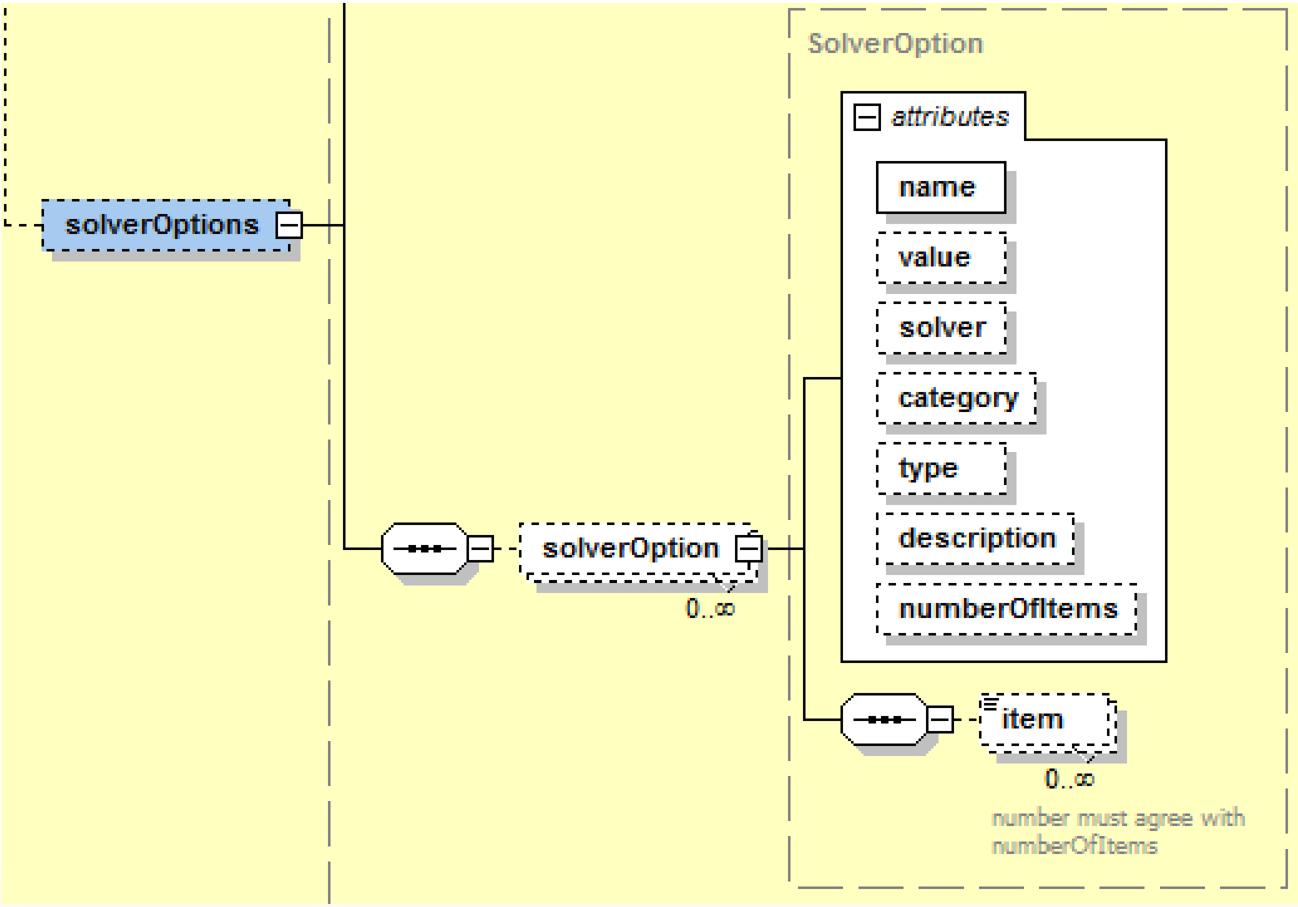


OSoL optimization schema element



© 2011 H.I. Gassmann

The solverOptions element



Sample .osol file

```
<?xml version="1.0" encoding="UTF-8"?>
<osol xmlns="os.optimizationservices.org"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="os.optimizationservices.org
    http://www.optimizationservices.org/schemas/2.0/OSoL.xsd">
    <optionHeader>
        <name>sample.osol</name>
        <source></source>
        <description>
            This file is intended as an illustrative example.
        </description>
        <fileCreator>
            Horand Gassmann, Jun Ma and Kipp Martin
        </fileCreator>
        <licence>
            This file is licensed under the Eclipse Public License.
        </licence>
    </optionHeader>
    <general>
        <solverToInvoke>couenne</solverToInvoke/>
        <serviceURI>
            http://74.94.100.129:8080/OSServer/services/OSSolverService</serviceURI>
        <instanceLocation locationType="http">
            http://myweb.dal.ca/gassmann</instanceLocation>
    </general>
```

Sampl .osol file (cont`d)

```
    <optimization>
      <variables>
        <initialVariableValues numberOfVar="2">
          <var idx="0" value="5."/>   <var idx="1" value="5."/>
        </initialVariableValues>
      </variables>
      <solverOptions numberOfSolverOptions="5">
        <solverOption name="print_level" solver="ipopt" type="integer" value="5"/>
        <solverOption name="max_iter" solver="ipopt" type="integer" value="2000"/>
        <solverOption name="tol" solver="ipopt" type="numeric" value="1.e-9"/>

        <solverOption name="LS_IPARAM_LP_PRINTLEVEL" solver="lindo"
              category="model"  type="integer" value="0"/>
        <solverOption name="LS_IPARAM_LP_PRINTLEVEL" solver="lindo"
              category="environment" type="integer" value="1"/>

        <solverOption name="node_limit" solver="couenne" type="integer"
              value="1000" category="bonmin" />
        <solverOption name="max_iter" solver="couenne" type="integer"
              value="2000" category="ipopt" />
      </solverOptions>
    </optimization>
</osol>
```

Transcription rules for in-memory representation

- XML complexType corresponds to C++ class
- XML element or attribute corresponds to member of C++ class
- XML sequence of identical elements corresponds to a C++ array

The OSoL schema – text version

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="os.optimizationservices.org" targetNamespace="os.optimizationservices.org"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="osol" type="OSoL"> </xs:element>
<xs:complexType name="OSoL">
    <xs:sequence>
        <xs:element name="optionHeader" type="GeneralFileHeader" minOccurs="0"/>
        <xs:element name="general" type="GeneralOption" minOccurs="0"/>
        <xs:element name="system" type="SystemOption" minOccurs="0"/>
        <xs:element name="service" type="ServiceOption" minOccurs="0"/>
        <xs:element name="job" type="JobOption" minOccurs="0"/>
        <xs:element name="optimization" type="OptimizationOption" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="OptimizationOption">
    <xs:sequence>
        <xs:element name="variables" type="VariableOption" minOccurs="0"/>
        <xs:element name="objectives" type="ObjectiveOption minOccurs="0"/>
        <xs:element name="constraints" type="ConstraintOption" minOccurs="0"/>
        <xs:element name="solverOptions" type="SolverOptions" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

DALHOUSIE UNIVERSITY
Inspiring Minds

Solver communication

- Goal: Avoid enumeration of supported options
- Depends on the solver API
- In principle scalar-valued options are tuples (usually name-value pairs)
- E.g. Ipopt:

```
optionsVector = osoption->getSolverOptions( "ipopt",true);
int num_ipopt_options = optionsVector.size();
for (int i = 0; i < num_ipopt_options; i++) {
    if(optionsVector[ i]->type == "numeric" )
        app->Options()->SetNumericValue(optionsVector[ i]->name,
            os_strtod( optionsVector[ i]->value.c_str(), &pEnd ) );
    else if(optionsVector[ i]->type == "integer" )
        app->Options()->SetIntegerValue(optionsVector[ i]->name,
            atoi( optionsVector[ i]->value.c_str() ) );
    else if(optionsVector[ i]->type == "string" )
        app->Options()->SetStringValue(optionsVector[ i]->name,
            optionsVector[ i]->value);
}
```

Results returned from solver

- Optimal variable values
- Optimal objective value
- Optimal dual values
- Range information
- Optimal basis information
- …
- Different solvers return different items in different formats
- Common syntax – individual semantics
- Same top level structure as OSoL
- Result of one optimization may be used as starting point for another

OSrL and OSResult

- Result of the optimization
  - Solution status
  - Statistics
  - Value of primal and dual variables
  - Basis information
- Can be displayed in a browser
- In-memory representation: **`OSResult`**
- API: **`get()`**, **`set()`**, **`add()`** methods

How to get OS

- Binaries
  - http://www.coin-or.org/CoinBinary/OS
    - OS-2.1.1-win32-msvc9.zip
    - OS-2.3.0-linux-x86-gcc4.1.2.tgz
    - OS-2.3.0-linux-x86_64-gcc4.3.2.tgz
- Stable source
  - http://www.coin-or.org/download/source/OS/
    - OS-2.4.1.tgz
    - OS-2.4.1.zip
- Development version (using svn)
  - svn co https://projects.coin-or.org/svn/OS/releases/2.4.1 COIN-OS
  - svn co https://projects.coin-or.org/svn/OS/trunk COIN-OS

QUESTIONS?

http://myweb.dal.ca/gassmann

http://www.optimizationservices.org

http://www.coin-or.org/projects/OS.xml

Horand.Gassmann@dal.ca