Optimization Services (OS)

Today: open Interface for Hooking Solvers to Modeling Systems

Jun Ma

Northwestern University

- Next generation distributed optimization (NEOS)
- Framework for Optimization Software Design
- Hosting Optimization/Computing as Services
- Standardizing representation/Communication

Joint with

Robert Fourer – Northwestern University

Kipp Martin – University of Chicago

at

DIMACS Workshop on COIN-OR

Rutgers University

07/19/2006

Sequence of Our Talks

- **An overview of Optimization Services and concepts (Jun Ma)**

- Using optimization services libraries as an interface for modeling systems (Bob Fourer)
- Using optimization services libraries as an interface for solvers (Kipp Martin)

Outline of My Part

- Motivation
- What is Optimization Services (OS) and Optimization Services Protocol (OSP)
- **“vs.” – fundamentals and clarifications**

Sharing our lessons and insights that we learned the hard way

- Current State of Optimization Services
- Derived and Future Research/Potential Collaboration
- Major User and Business Values

Motivation

1. Tightly-coupled implementation
2. Various operating systems
3. Various communication/interfacing mechanisms
4. Various programming languages
5. Various benchmarking standards

- **The key issue is communication (includes interfacing), not solution!**
- **… and Optimization Services is intended to solve all the above issues.**

Motivation

- **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.

Motivation

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.

be

constraint programming

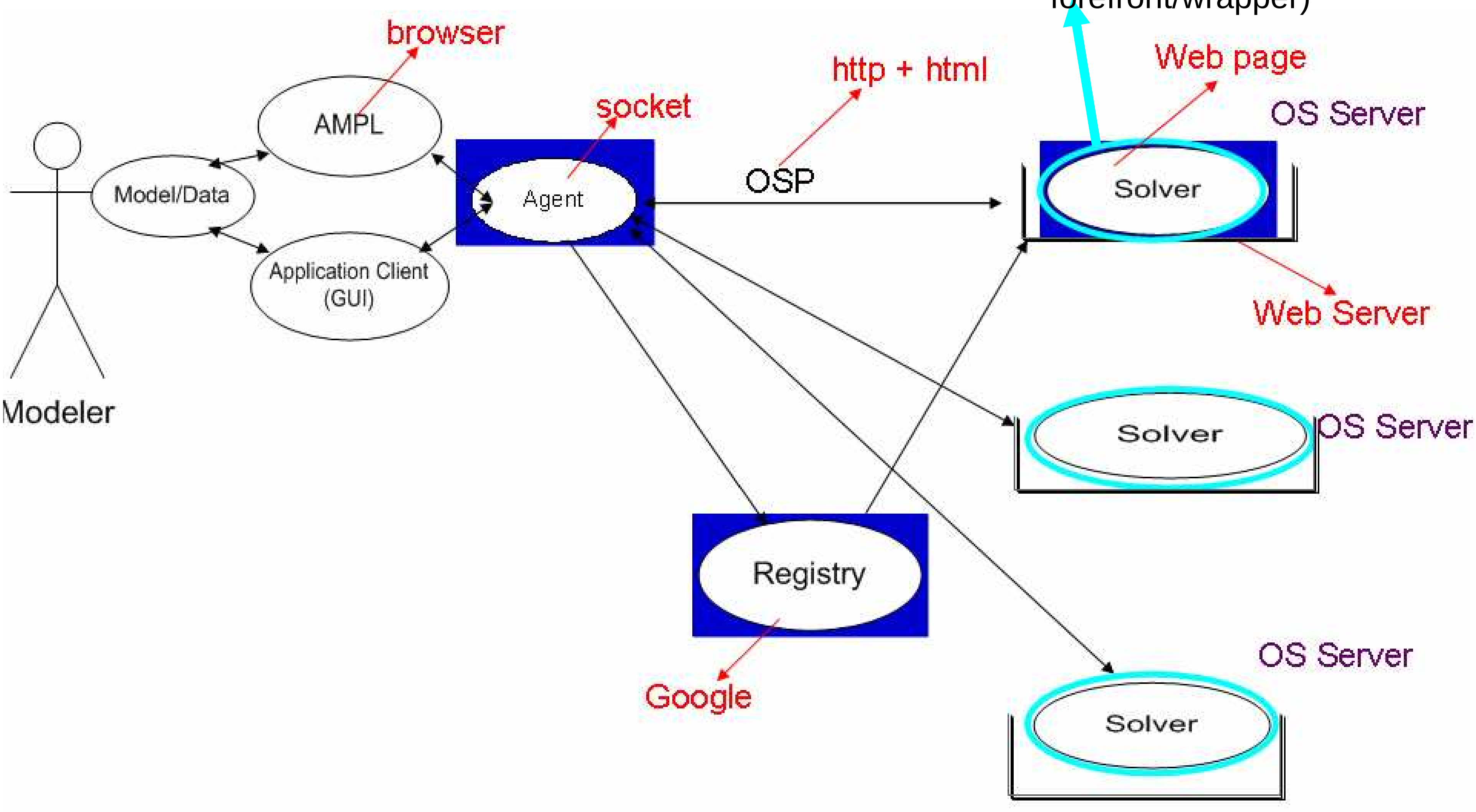
Corollary 2: Optimization should available as a software service. It should be easy to solve optimization problems of any type (linear, integer, nonlinear, stochastic, etc), at any time, if you are hooked up to the network.

Optimization Services is our attempt to make optimization a service.

What is Optimization Services (OS) and Optimization Services Protocol (OSP)

What is OS and OSP

Service (public forefront/wrapper)



What Does Optimization Services Do

- Provide a set of standards and protocols for distributed optimization (local as a special case)
- Provide protocols for representing problem instances, solution instances, and option instances
- Provide protocols that facilitate (stateful) communication (synchronous and asynchronous) between solvers and clients that use the solvers
- Provide protocols allow clients that use optimization solvers to discover their existence over the network and allow solvers to register their existence
- Provide libraries that can be used in a modeling environment to read, write, communicate model instances and solutions

“vs.” – fundamentals and concept clarifications

Either you don’t see
Or it’s so obvious

-- Michael Henderson

But always easy to forget

-- Jun Ma

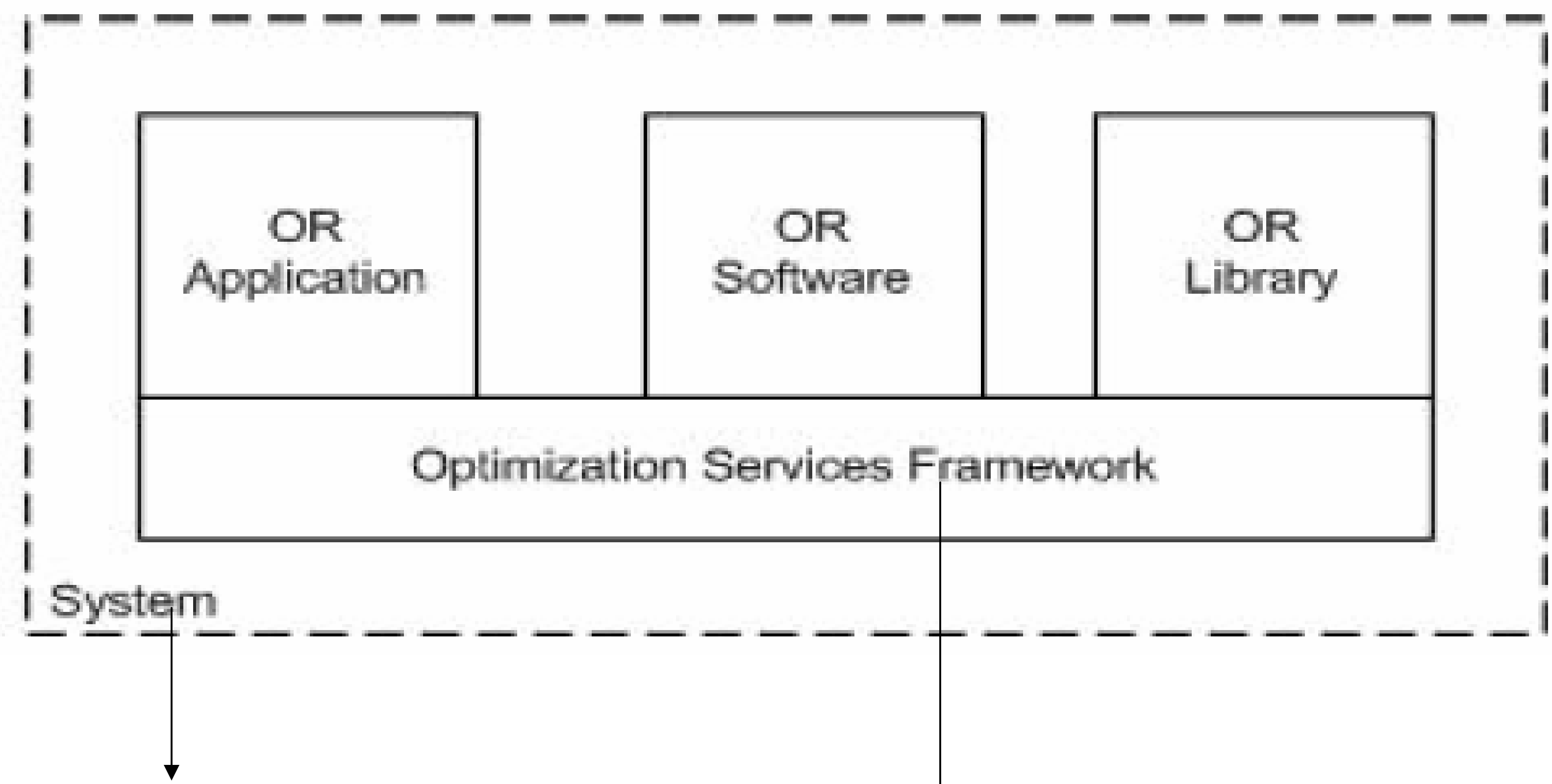
Standardizing vs. Not Standardizing

- Clarify what OS tries to standardize and
  what OS does not do
- OS is intended to be a universal framework …
- … there are many different reasons for making the decision not to standardize certain things
- e.g. Model vs. Instance
- e.g. Solver vs. Instance

Design vs. Implementation

- Optimization Services is more about design
- 70% vs. 30% is the current state
- 90% vs. 10% is at least my goal
- The current OS design is the result of thinking about everything at the same time from scratch rather than sequentially (LPFML is subsumed by OSiL)
- A good design takes time but usually looks natural and simple after it is finished.
- A good design is extendable, but we will be on the very conservative when it comes to extending the core.

Framework vs. Library



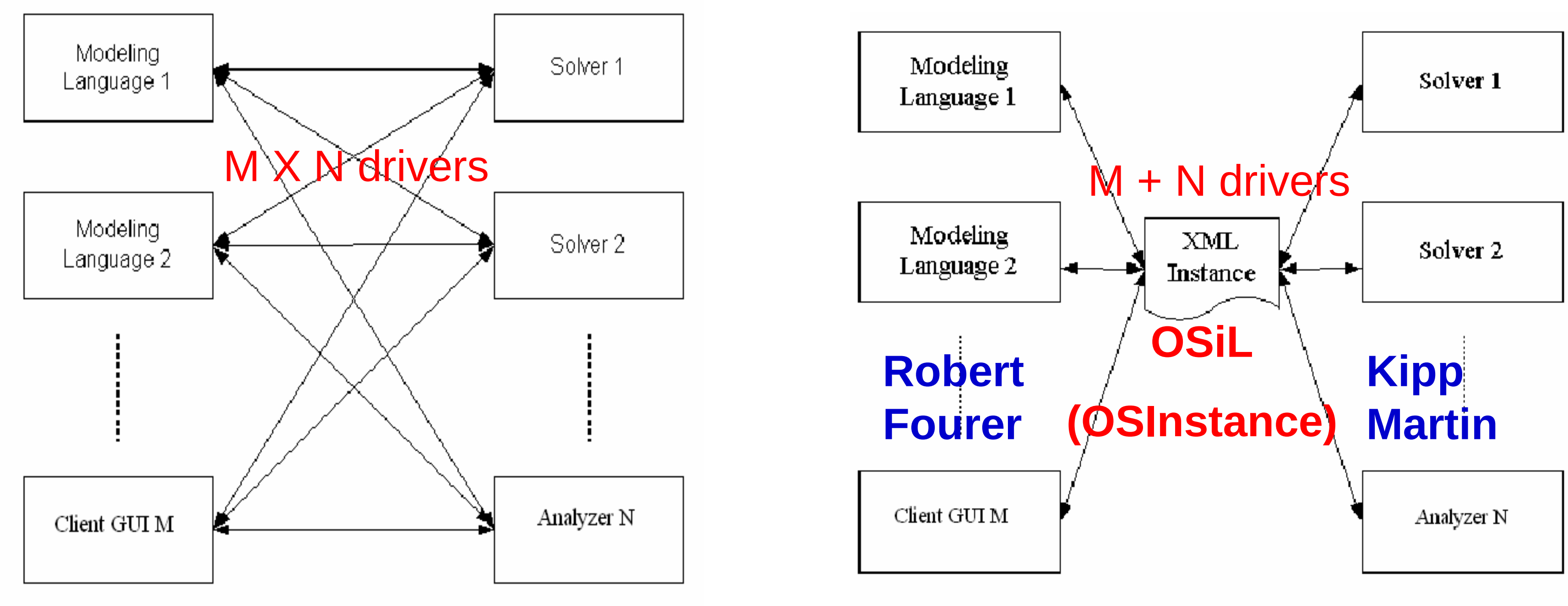
Implementation

(c++, java, .net)

Design

Think of Sun J2EE (now Java EE 5) and IBM Websphere

Model vs. Instance



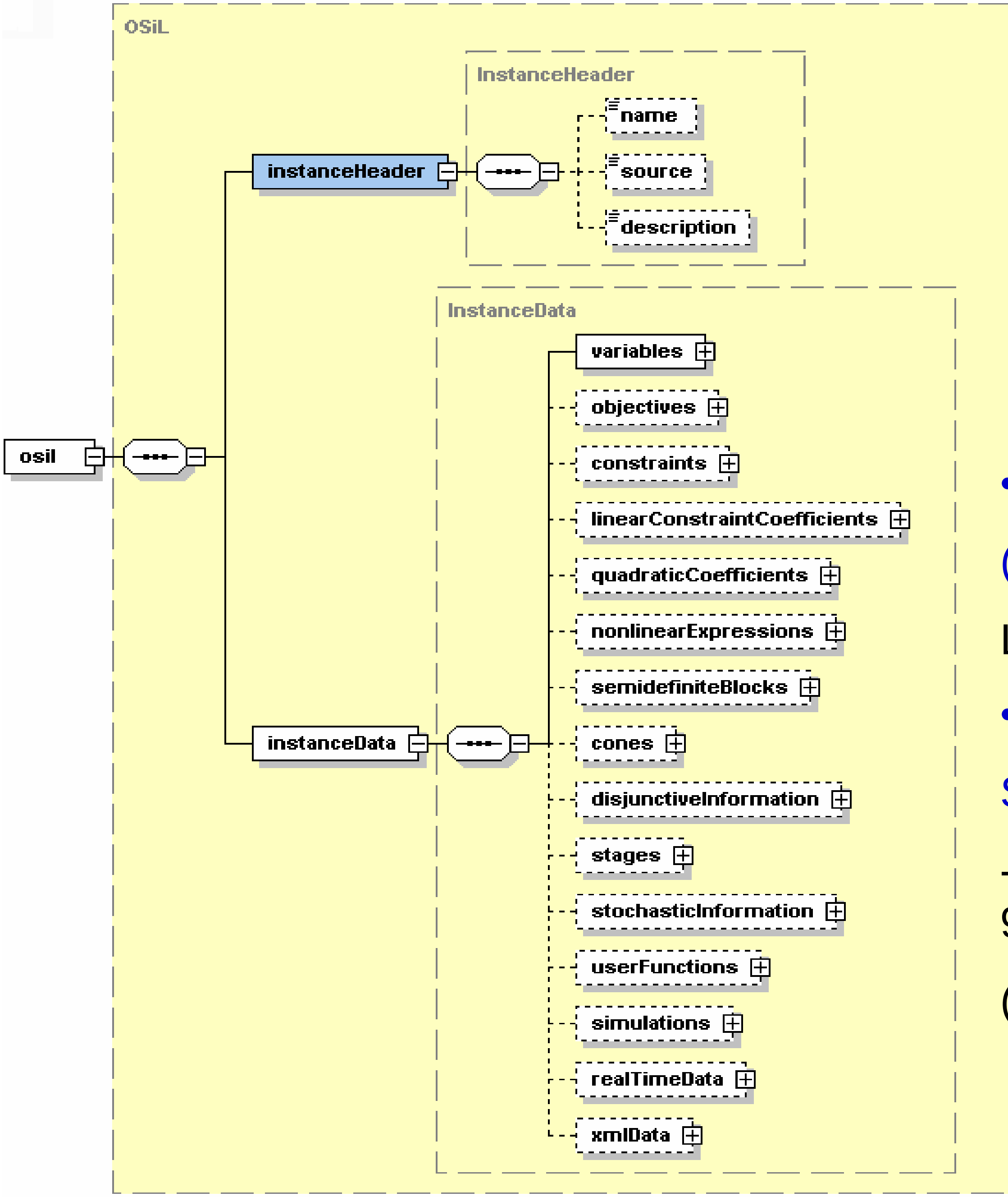
Model: AIMMS, AMPL, GAMS, LINGO, LPL, MOSEL, MPL, OPL, OSmL, POAMS, PuLP, spreadsheets, GUIs

Instance = Model + Data: MPS, xMPS, LP, LPFML, SIF, SDPA, .nl (AMPL), instruction list (Lingo)

Optimization Services doesn't standardize model

(Think of user-friendliness, Java/byte code, .net/MSiL)

Instance vs. solver



- Instance (what OS standardizes)

Optimization Services instance Language (OSiL) – string/file &

OSInstance – in-memory object/data strcutre

- Solver API

(What OS does NOT standardize)

Lindo API, Concert, OSI, Impact API

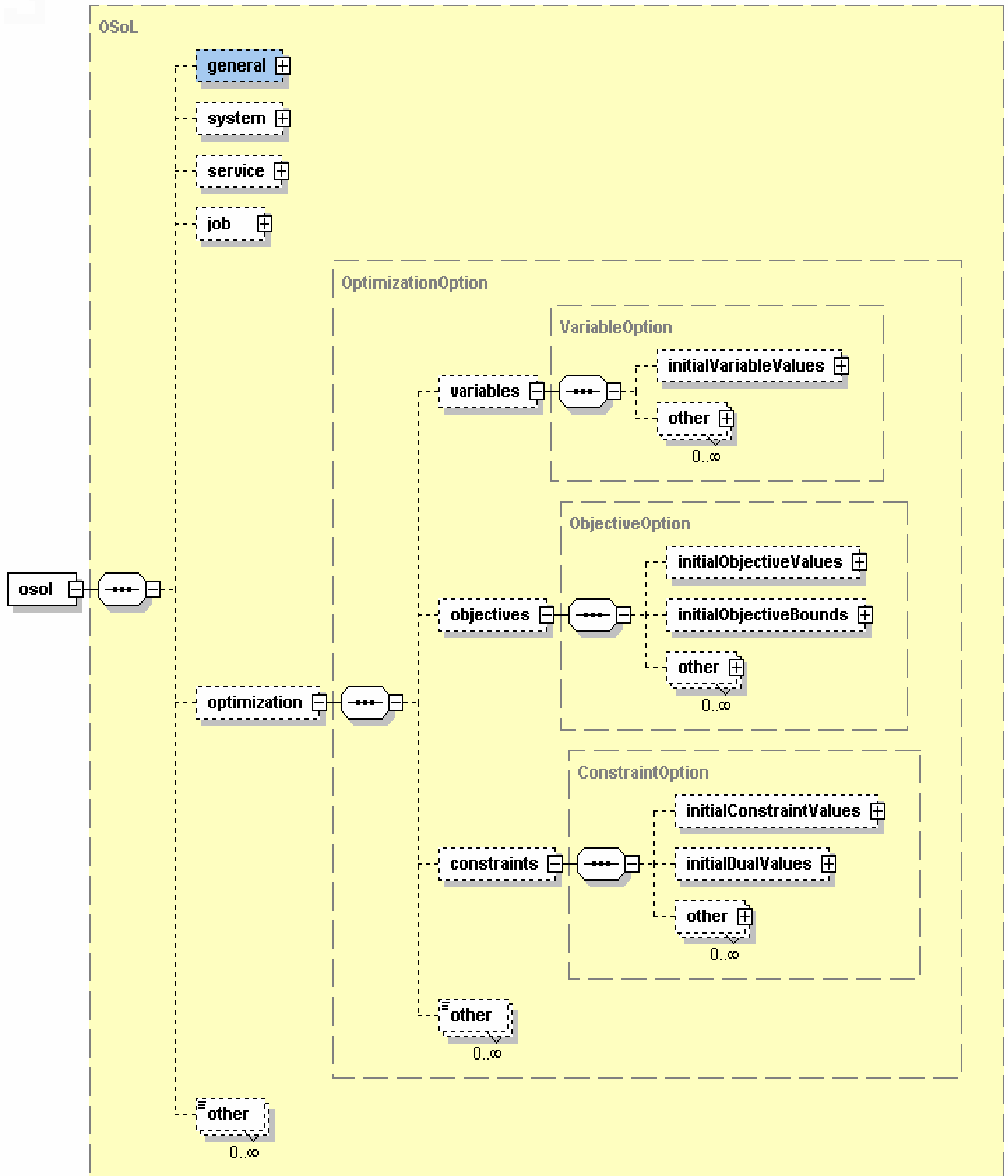
- Standardizing Solver API ==

Standardizing API for algorithms

-- standardizing instance can save 90% of the time of code

(think of SQL vs. JDBC/ODBC)

Instance vs. Option



- Instance vs. Option ==

  Instance vs. Solver/Algorithm

- solveMIP == solve() + option->MIP
- solveGOP == solve() + option->global
- initialSolve == solve() + option->init
- similar for other methods

OS does have a format for option

But mainly for services

Hardly any for optimization

Use <other> elements

Methods vs. Arguments

initialSolve(String OSiL, String OSoL)

solveMIP (OSInstance instanceObject)

setOSInstance(OSInstance instanceObject)
setOSOption(OSOption optionObject)
minimizeUsingBranchAndBound( )

OS standardizes the arguments not Solver Methods

Think about ODBC/JDBC and SQL

ODBC/JDBC is about database connectivity and is the method part (how). Methods include execute, update, delete, etc. etc. etc. etc.

SQL is the query to be sent to the database through ODBC/JDBC and is the argument part (what). “select *.* from … where … “

Communication vs. Representation

Communication is similar to "Methods" (how)
Representation is similar to "Arguments" (what)

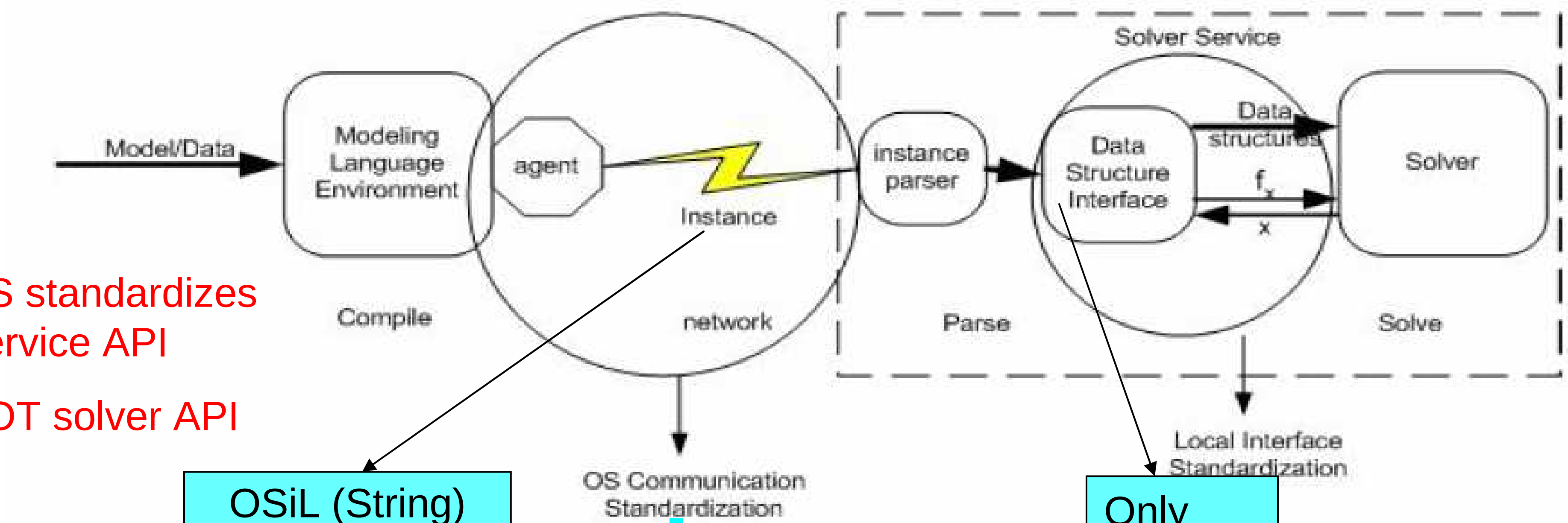
HTTP is the communication
HTML is the representation
HTML is sent via HTTP

OS standardizes both communication and representation

Communication are the methods on services (OShL)

Representation is about what to communicate (OSiL, OSoL, OSrL, OSpL etc. – only strings not objects)

Services vs. Solver



OS standardizes Service API

NOT solver API

OSiL (String)

OShL (hook)

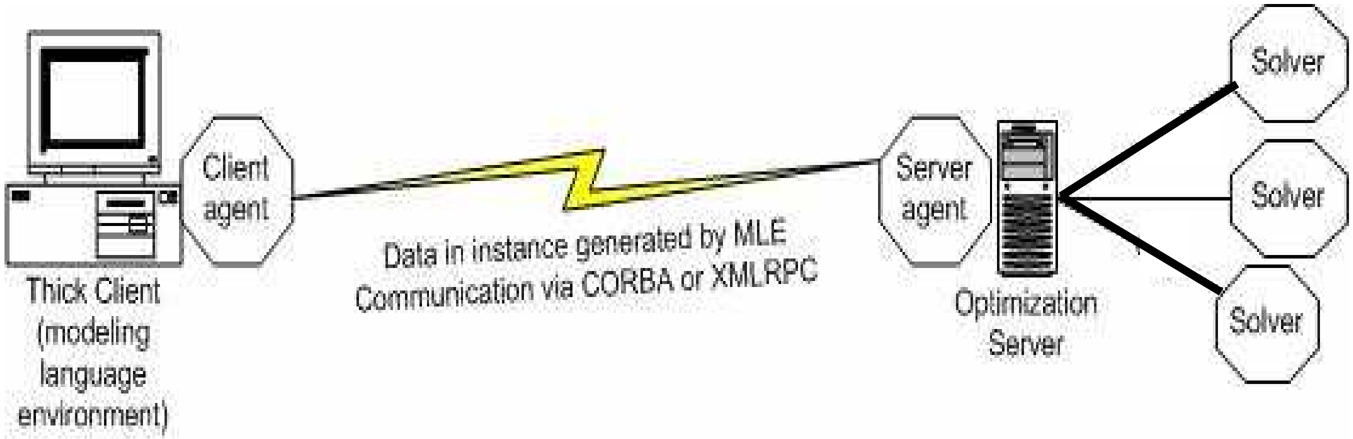
Only instance

getJobID(OSoL) → jobIDString	//key in maintaining state
solve(OSiL, OSoL) → OSrL	//synchronous call
send(OSiL, OSoL) → true/false	//asynchronous call
retrieve(OSoL) → OSrL	//coupled mainly with send
kill(OSoL) → OSpLOutput	//different from “stop” on IE
Knock(OSpLInput, OSoL) → OSpLOutput	//intermediate process info

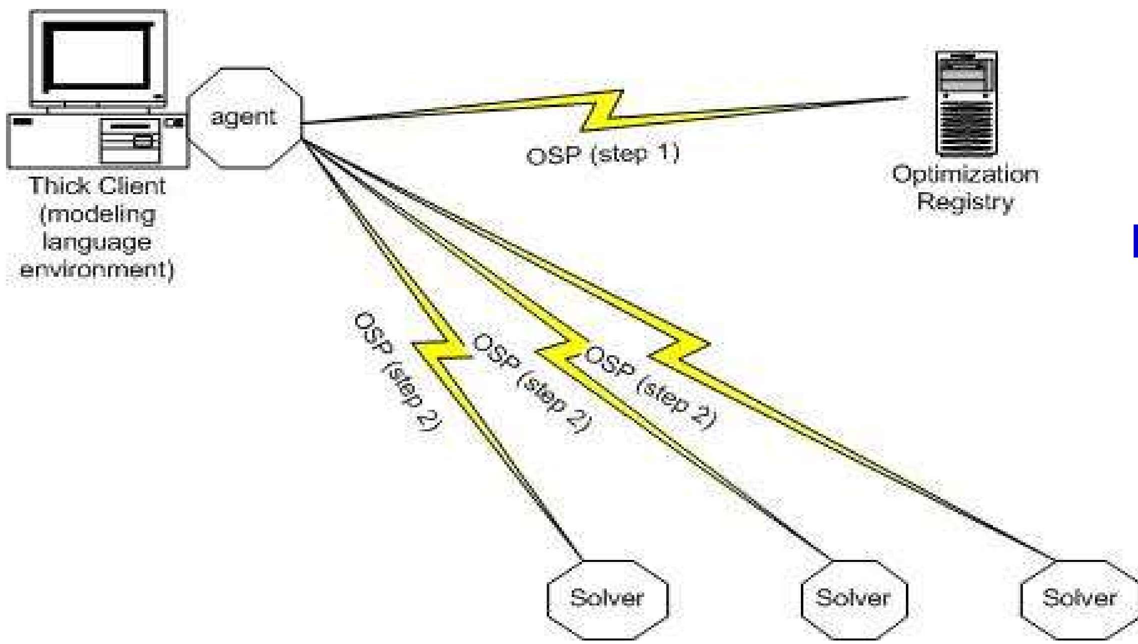
Distributed vs. Local

- Optimization Services partially started as a next-generation NEOS project; therefore with “distributed” in mind
- But you always start from a local computer and sooner or later it will reach the other local computer
- “Local” is a derived research of Optimization Services and is what the next two talks about for today

Centralized vs. Decentralized



Centralized

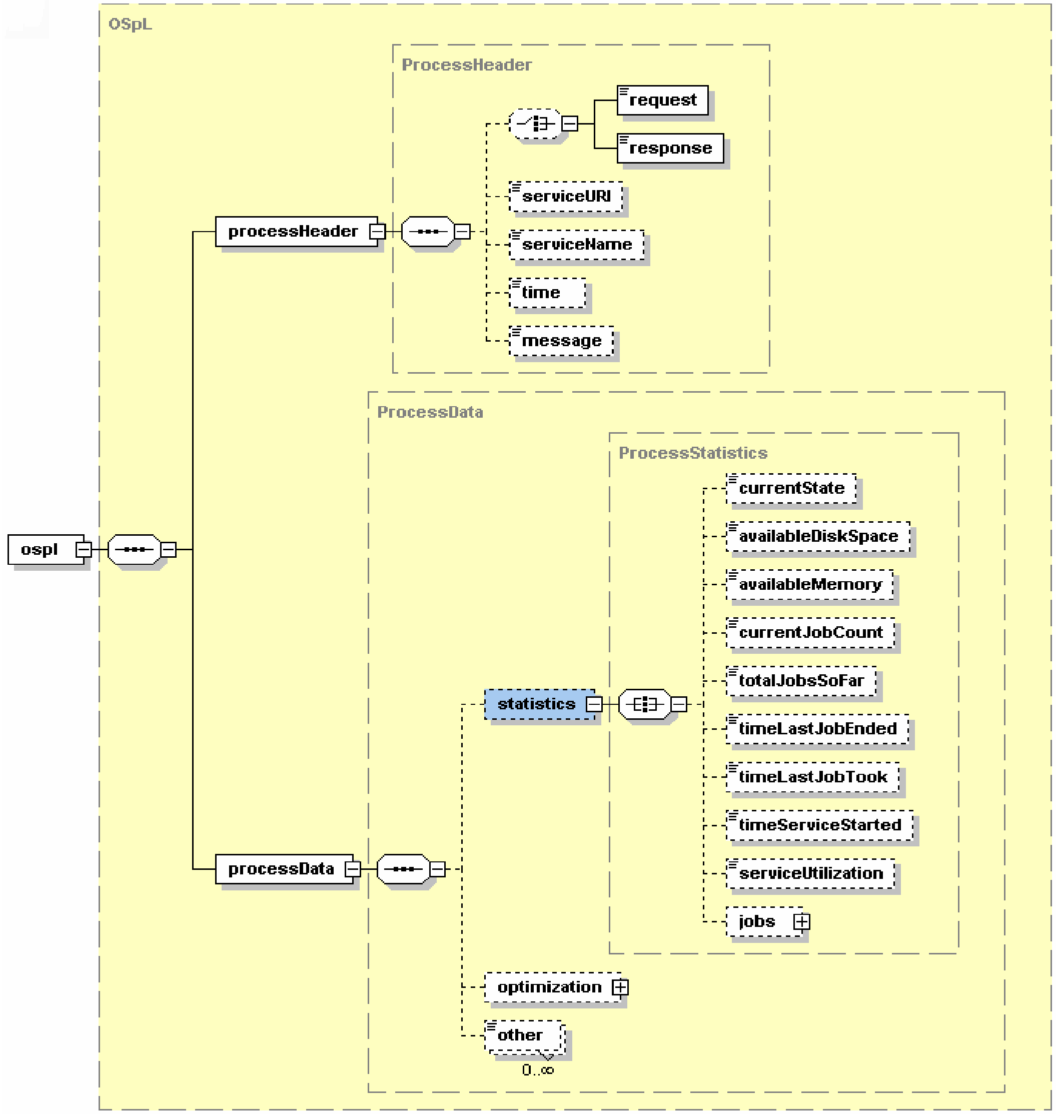


Decentralized

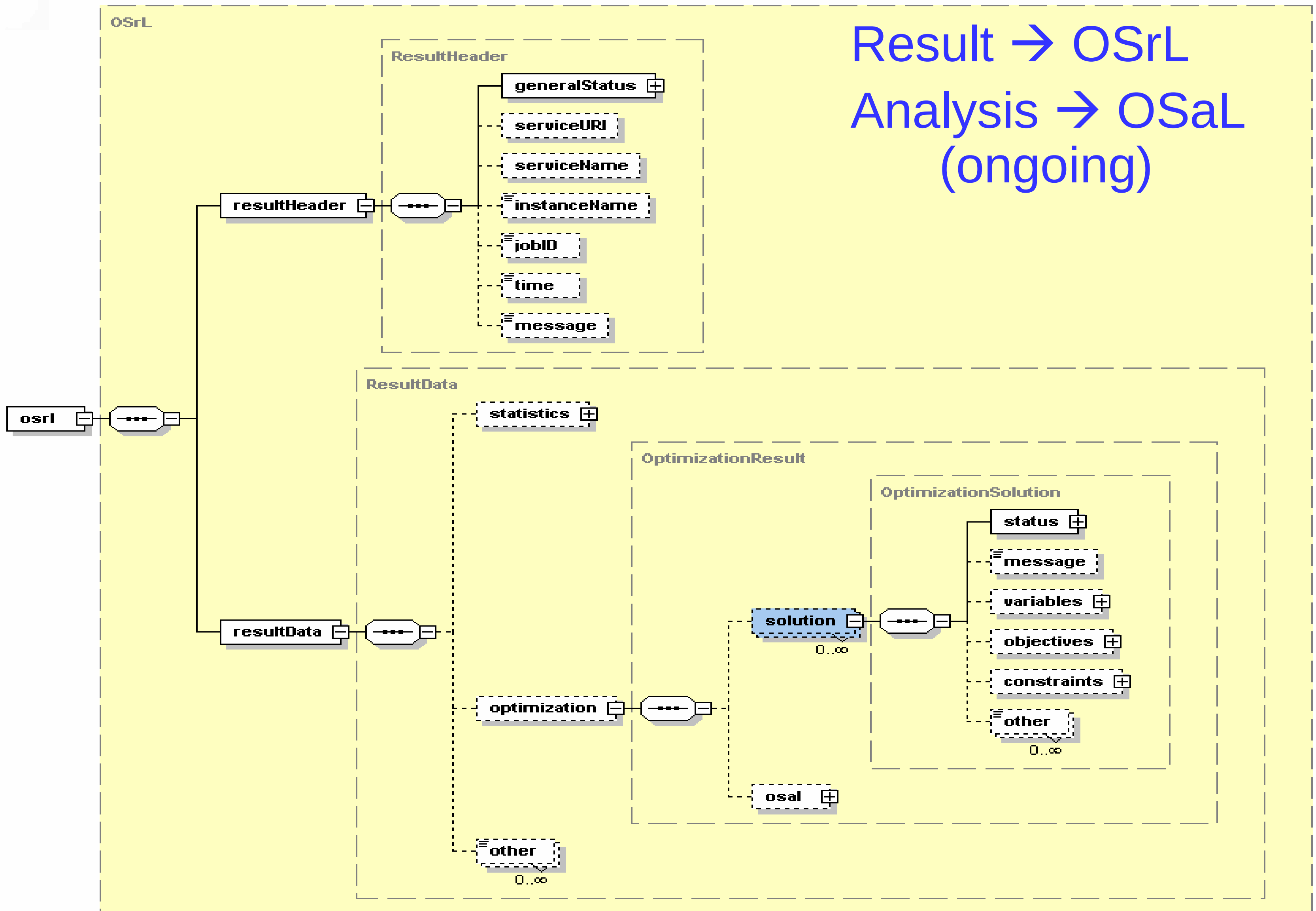
Result vs. Process

- Result → the final output of solver/service
- OS does Standardize (OSrL)

- Process → Intermediate Result? (OSpL)
- OS mainly standardizes the service part, only a little on the optimization part



Result vs. Analysis



Result → OSrL

Analysis → OSaL (ongoing)

State of Optimization Services

- Implementation in C++, Java, .net
- Instance Extension (core is linear)

Integer, nonlinear, constraint programming user-functions, optimization via simulation, real-time, ongoing: disjunctive, cone, semidefinite, (Fourer, Ma, Martin)

GlobalOptimization? NetworkandGraph?

Stochastic Extension (Fourer, Gassmann, Ma, Martin)

- Modeling Languages

OSmL [native] (Ma, Martin)

AMPL (Fourer, Ma, Martin)

Lingo (Ma, Martin, Shrage //todo)

Spreadsheet, GAMS //planned)

- Solvers

IMPACT [native] - convex/GMIP/Parallel (Ma, Mehrotra, Sheng)

CPLEX, Knitro, Lindo, CLP, CBC, CLP, GLPK, planned IPOPT, etc. (Ma, Martin, Sheng)

- Framework/System (commercial deployment, and noncommercial (NEOS planned)
- Licenses (CPL)
- Third party (Leo Lopez etc.)

OS Repository

Netlib, Kenninton, Infeasible_set, MIP 2003

COPS (planned)

-- all represented in OSiL

-- all well documented

OS Web pages (www.optimizationservices.org under construction, soon mutually linked with COIN-OR.org)

OS Repository and Documents will be available from both sites.

Derived and Future Research/Potential Collaboration

- Chapter 10 Future Work and Derived Research from Optimization Services
- 10.1 The Optimization Services Project
- 10.2 Standardization
- 10.3 Problem Repository Building
- 10.4 Library Building
- 10.5 Derived Research in Distributed Systems
- 10.6 Derived Research in Decentralization
- 10.7 Derived Research in Local Systems
- 10.8 Derived Research in Optimization Servers
- 10.9 Derived Research in Computational Software
- 10.10 Derived Research in Computational Algorithms
- 10.11 Commercialization and Derived Business Models

User Experience

- Open Environment
- Convenience just like Using Utility Services
- No High Computing Power Needed
- No Knowledge in Optimization Algorithms and Software (solvers, options, etc.)
- Better and More Choices of Modeling Languages
- More Solver Choices
- Solve More Types of Problems
- Automatic Optimization Services Discovery
- Decentralized Optimization Services Development and Registration
- More Types of Optimization Services Components Integrated (Analyzers/Preprocessors, Problem Providers, Bench Markers)
- Smooth Flow and Coordination of Various Optimization Services Components.
- A Universal, Scalable and Standard Infrastructure that promotes Collaboration and Other Related Researches
- Concentration on Good Modeling

Business Values

Solve more types of computational problems more efficiently

Easily deploy enterprise computing system within a company, with intelligent components in scheduling computational jobs, registering and finding computing services, routing maintenance

Provide computational software as services on dedicated servers

Let all computational software communicate with each, independent of platforms and implementations.

Save costs on expensive software licenses

Make full use of limited computational solvers