Optimization Services (OS) Framework and OSP Protocols (OSxL)

"Combining Operations Research with Computing Technology"

Robert Fourer

Jun Ma

Northwestern University

Kipp Martin

University of Chicago

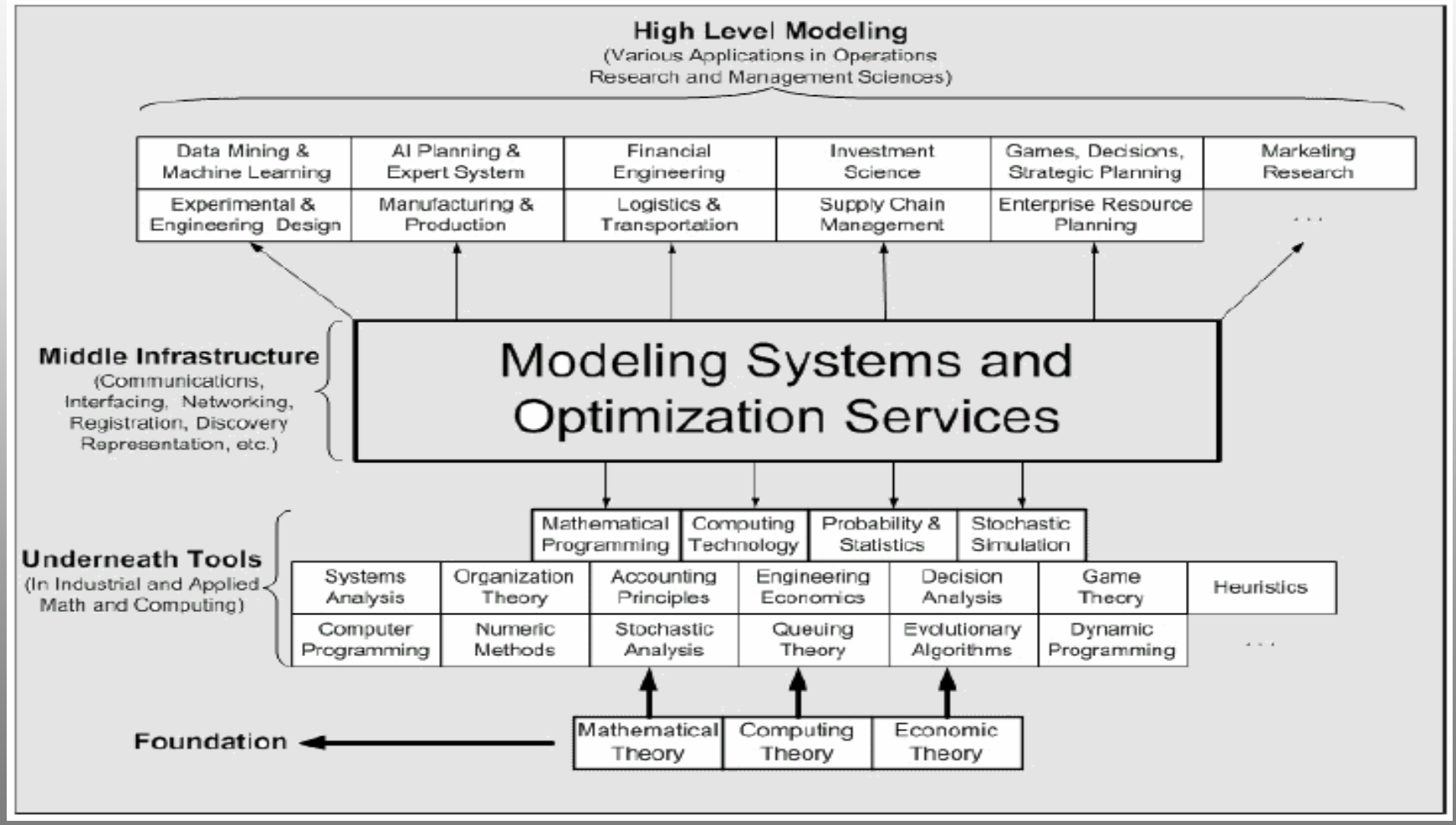
Jun Ma

maj@northwestern.edu

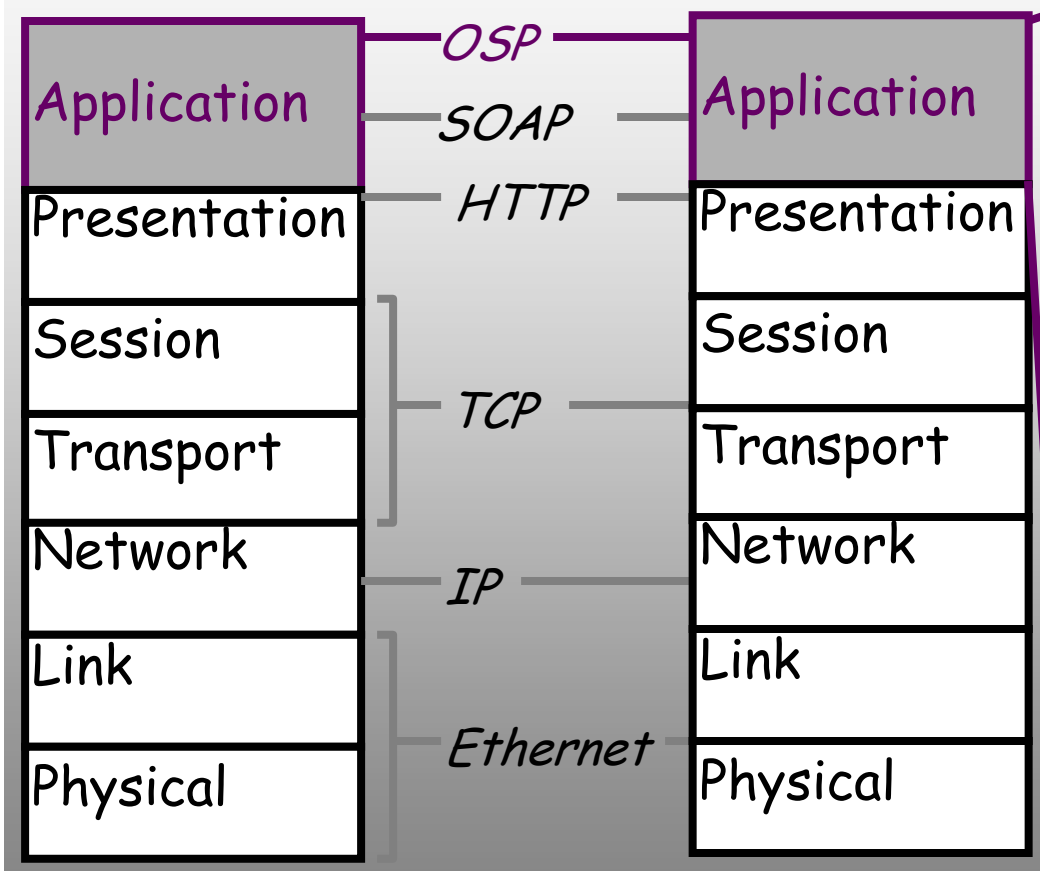
10/24/2004

INFORMS Conference, Denver

The Positioning of Optimization Services Framework in OR/MS



The Positioning of OSP Protocols (OSxL) in Computing



The 7-layer OSI Model

The 4-layer Internet model

```
GET /xt/services/ColorRequest HTTP/1.0
Content Length: 442
Host: localhost
Content-type: text/xml; charset=utf-8
SOAPAction: "/getColor"
```

```
<soap:Envelope>
  <soap:Body>
```

OSP – specifies soap content

Communication Interface Representation

e.g. hook (“<OSiL> … </OSiL>”)

```
  <soap:Body>
</soap:Envelope>
```

Optimization Services (OS) Framework

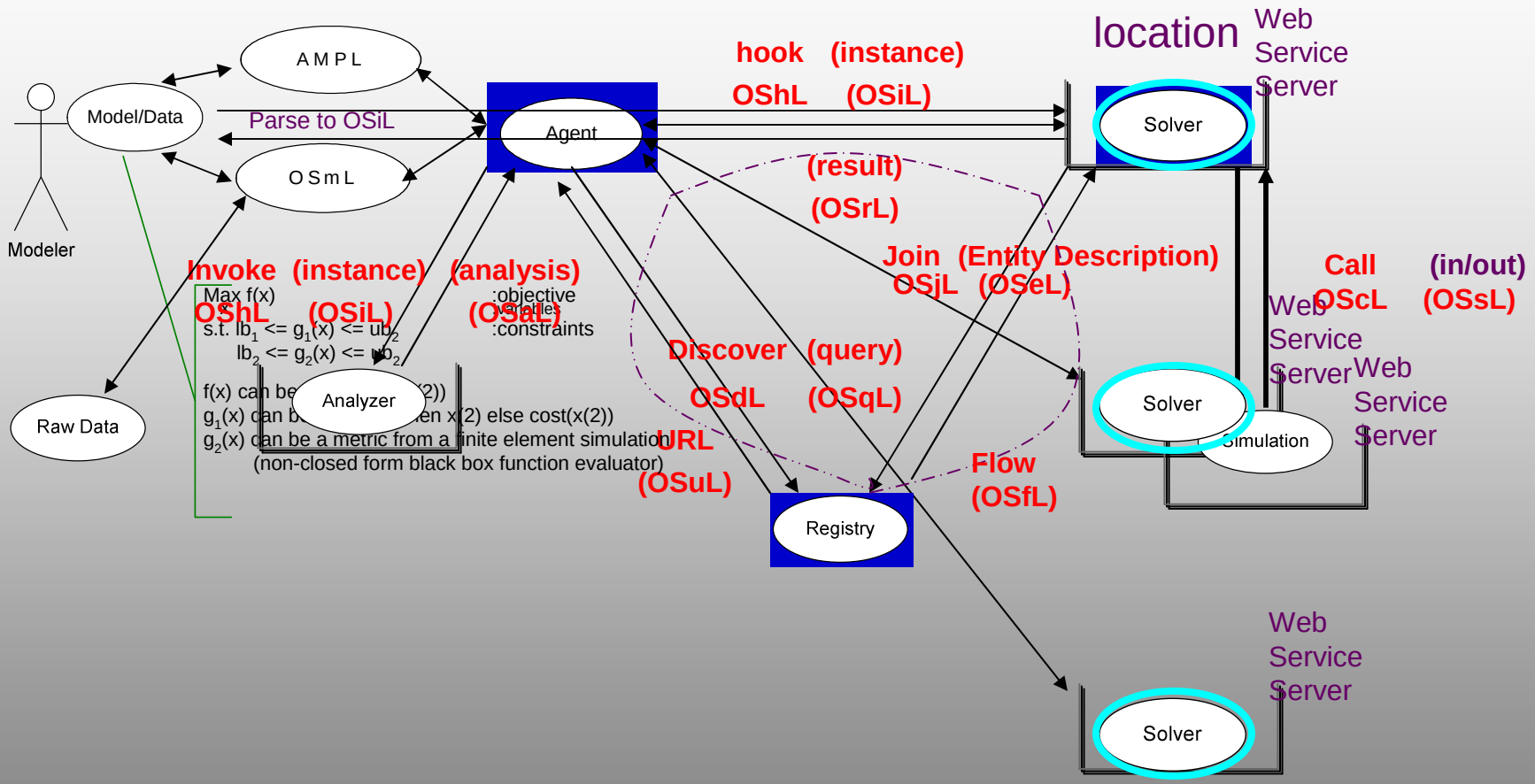
- A framework, NOT a system
  - cf. constitution, NOT government/Court System. Only that the framework specifications are written in XML languages (NOT English).
  - cf. DOM/SAX, NOT Xerces, Crimson, or other real implementations/providers.
  - But we are in the middle of developing the modeling system according to this framework.
  - We are also building libraries for other people to put up their optimization services.
- Distributed environment (Local environment being just a special Case)
- Service Oriented Decentralized Architecture (Registry NOT Server).
- Optimization Services Components

1. Modeling Language Environment (MLE) (e.g. AMPL, OSmL) -- OSModel
2. Optimization Registries (e.g. The next generation NEOS) – OSRegistry
3. Analyzers/Preprocessors (e.g. Mprobe, Dr. AMPL) -- OSAnalyzer
4. Optimization Solvers (e.g. Lindo) -- OSSolver
5. Simulation (e.g. Finite Element Analysis) -- OSSimulation
6. Communication Software Agent – OSAgent
7. All of the above are communicating in a common language -- OSCommon

XML-based standard

Optimization Services (OS) Framework

The next generation NEOS
THE Optimization Internet
the Universal distributed/local COIN for OR



[Standard, Simple, Scalable] => Smooth

- **The General and Universal Framework for Optimization in Local and Distributed Environment.**
- **Combining Optimization with Modern Computing Technologies.**
- **A Next Generation Modeling System as An Internet Resource.**
- **Standardization of Optimization Representation, Communications, Registration, and Discovery**
- **Using Optimization Computing Tools Just Like Daily Utility Services.**

Chapter 12. OS Representation

1. OSlL (linear) – in honor of the original LP-FML
2. OSgL (general) – general schema
3. OSnL (nonlinear nodes) – nonlinear node definitions
4. OSiL (instance) – optimization instance
5. OSrL (result) – optimization result
6. OSoL (option) – solver option
7. OSaL (analysis) – analyzer metadata
8. OSsL (simulation) – simulation engine input and output
9. OStL (template) – template holding other representations
10. OSmL (modeling) – XML query based modeling language

Chapter 13. OS Communication

1. OScL (call) – call simulation engines
2. OShL (hook) – hook up solvers
3. OSdL (discover) – discover optimization services
4. OSjL (join) – join OS registries
5. OSfL (flow) – orchestrate flow of OS invocations
6. OSvL (validate) – validate OS representations

Chapter 14. OS Registration and Discovery

1. OSeL (entity) – endpoint OS component static description
2. OSpL (process) – OS component runtime process description
3. OSzL (zero) – dummy instance for sending signals
4. OSqL (query) – query language for OS components
5. OSuL (URL) – query result containing OS component URL addresses
6. OSyL (yellow pages) – organization of registry information
7. OSbL (benchmarking) – benchmark information of OS solvers
8. OSkL (knowledge) – knowledge template holding other component information
9. OSwL (web page) –XSLT for standard web publication of OS components

User Experience Movement

- 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 University, Scalable and Standard Infrastructure that promotes Collaboration and Other Related Researches
- Concentration on Good Modeling

Why Not MathML

"Need." Content MathML includes more than we need in the OR/MS community. If an instance unintentionally includes these features which shouldn't be allowed, MathML DTD or Schema would still validate while none of the solvers would ever recognize such features.

"Design." OSnL has a very consistent recursive design There is also a one -to-one correspondence between each node element in the expression tree and each node class in the parsing library. Content MathML cannot achieve the consistency because it has to tailor to general needs. The <apply> element in Content MathML consistently result in lengthier representation of nonlinear functions than OSiL.

"Specialty" We have all the special features such as XPath node, user functions and parameters, variable subscripts supported. Content MathML can work around some, but in awkward ways.

"Level." There is a reason that Content MathML is not called "Computation MathML." MathML is at a different level of the "bottom", maybe higher because it's still intended for "symbolic" content representation. Content MathML is a content-faithful transformation from the high level in that Content MathML retains original content. OSiL may be more appropriately called "Computation MathML." For example OSiL does substitution for high level identifiers and it's a "numerical instance" at the bottom, which no longer retains all the information of the original model.

"Control." The OR community does not have control over the design of MathML. Certain features that are critical in optimization may not be "naturally" built or not supported at all in MathML. As long as one feature is not supported at all in Content MathML, we should not adopt it.

"Flexibility." We can embed MathML in OSnL and OSiL, while MathML does not embed OSxL.

"Optimization." Critical optimization related information are treated more importantly than other general math information.