Ontology Design for Solver Reuse in Web Services Based Model Management Systems

Keun-Woo Lee[a] and Soon-Young Huh[b]

[a] *Graduate School of Management, Korea Advanced Institute of Science and Technology*
207-43 Chongyang-ni, Dongdaemoon-gu, Seoul 130-722, South Korea
Tel: +82-2-958-3650, Fax: +82-2-958-3604, E-mail: kwlee@kgsm.kaist.ac.kr

[b] *Graduate School of Management, Korea Advanced Institute of Science and Technology*
207-43 Chongyang-ni, Dongdaemoon-gu, Seoul 130-722, South Korea
Tel: +82-2-958-3626, Fax: +82-2-958-3604, E-mail: syhuh@kgsm.kaist.ac.kr

Abstract

As complex mathematical models are increasingly adopted for business decision-making, difficulties arise in reusing solvers (i.e., model solving algorithms) against diverse models and data sets and thus the collaboration among users (model/solver builders and decision makers) in multiple departments becomes very difficult. To facilitate the solver reuse, this paper adopts the Web services technologies as the base technologies for linking the solvers to the models, both of which are created on different modeling paradigms and different system platforms, in unified system architecture. Specifically, this paper focuses on designing an ontology that represents the interfacing semantics of the model-solver interactions in a general and standardized form. By referring to the ontology, a model management system (MMS) can autonomously suggest a set of compatible solvers and apply them to individual models even though the decision makers are not knowledgeable enough about all the details of the models and the solvers. Thus, this Web services based MMS would improve the reusability of the solvers by relieving the decision makers from the risk of erroneous application of a solver to syntactically and semantically incompatible models and the burden of considerable understanding of model and solver semantics.

Keywords:

Model Management System, Solver Reuse, Ontology Design

Introduction

As business environments become more competitive and rapidly change, decision support systems (DSS) for precise and agile decisions have been increasingly adopted in many organizations. In a DSS, for user-friendliness and intuitive solution, a decision problem is formulated as a model, and for solving the models, diverse sets of solvers, i.e., model-solving algorithms, are also provided. Specifically, model management system (MMS) is dedicated to manage the entire life cycle of models as a part of the three component modules of a DSS together with a database management system (DBMS) and dialogue management system [12].

In the model management research area, there have been a wide spectrum of studies ranging from the modeling languages for effective modeling of management science/operations research (MS/OR) problems [4,7,8], the model representation scheme for easier creation, retrieval, and execution of models [8,10], to the DSS component integration framework for the reuse of models with different data sets for different problems [3,5,11]. However, there exist little effort focusing on the precise and flexible decision making support by the reuse of solvers [10] against diverse models and data sets. The solver reuse means that a single solver can be applied to multiple models having similar problem structures and a model can be solved using multiple solvers depending on the problem-solving purposes. Such flexible solver reuse can make the decision support process to be more user-friendly, streamlined, and result in a more simplified system architecture that leads easier implementation and maintenance of DSS.

To attain these benefits from the solver reuse sufficiently, a user of the DSS should be knowledgeable about which solvers can be applied to the model. Also, once an applicable solver is chosen, the user should understand how to match the individual model parameters to the solver parameters to execute the solver adequately [10]. In reality, however, since the semantic understanding of a model or a solver is not a trivial task, ordinary users, even sometimes professional users, often have difficulty in picking out the applicable solvers from the organizational solver library and executing them by matching the model parameters to the solver parameters [1,6]. Moreover, as more organizations have constructed the DSS distributed across their internal/external networks [9], models and solvers of

an organization have been created based on different modeling paradigms and different system platforms. Thus, the MMS, as a dedicated tool for managing the models, is required to support the following two capabilities to make the solver reuse easy and productive. First, for a specific model under consideration, the MMS should be able to suggest autonomously a set of solvers that are both syntactically and semantically compatible with the model. Second, when a particular solver is chosen for the model, the MMS should be able to match the model parameters to the adequate solver parameters intelligently and produce the model solving results even though the user cannot perform exact matching between the two.

Recognizing such requirements of the MMS, this paper focuses on the development of an interfacing ontology managing the interacting semantics between individual models and their compatible solvers for facilitating the autonomous solver suggestion and intelligent model solution capabilities. The interacting semantics include the compatibility of a solver with a model and their parameter matching patterns. Based on the interfacing ontology, two software agents, model and solver agents, are defined to assist a user's model-solving activities. Without direct users' intervention, the agents suggest the compatible solvers for a given model and match the model parameters with the solver parameters [2] by referring to the interfacing ontology. Specifically, in developing the ontology and the agents, we adopt the Web services technologies [13] as a vehicle for integrating the models and solvers distributed across networks and created based on different modeling paradigms and system platforms. The Web services technologies have been highlighted as a way of integrating distributed and heterogeneous applications, which was previously impractical because of non-interoperable proprietary approaches. In this paper, individual solvers are encapsulated as Web services (called solving services) that provide the model-solving functions.

Conceptual Architecture of Web Services Based MMS

Figure 1 shows the conceptual architecture of the Web Services based MMS proposed in this paper. The architecture has two primary tasks: management of the interfacing ontology and execution of the solver suggestion and model solution. To support these two tasks, a two-layered approach is provided. On the lower layer, the interfacing ontology exists to manage the interfacing semantics between the models and solvers. On the upper layer, the model and solver agents are placed to perform the autonomous solver suggestion and intelligent model solution. In conducting the tasks, the agents specifically look up the interfacing ontology and resolve any possible conflicts among the models and solvers caused by their different modeling paradigms and system platforms. The interfacing ontology and the two agents are to be explained in the following sections.

Interfacing Ontology

As mentioned earlier, for the flexible solver reuse including the autonomous solver suggestion and intelligent model solution, the interfacing ontology manages the interacting semantics between individual models and their compatible solving services. The interacting semantics specifically

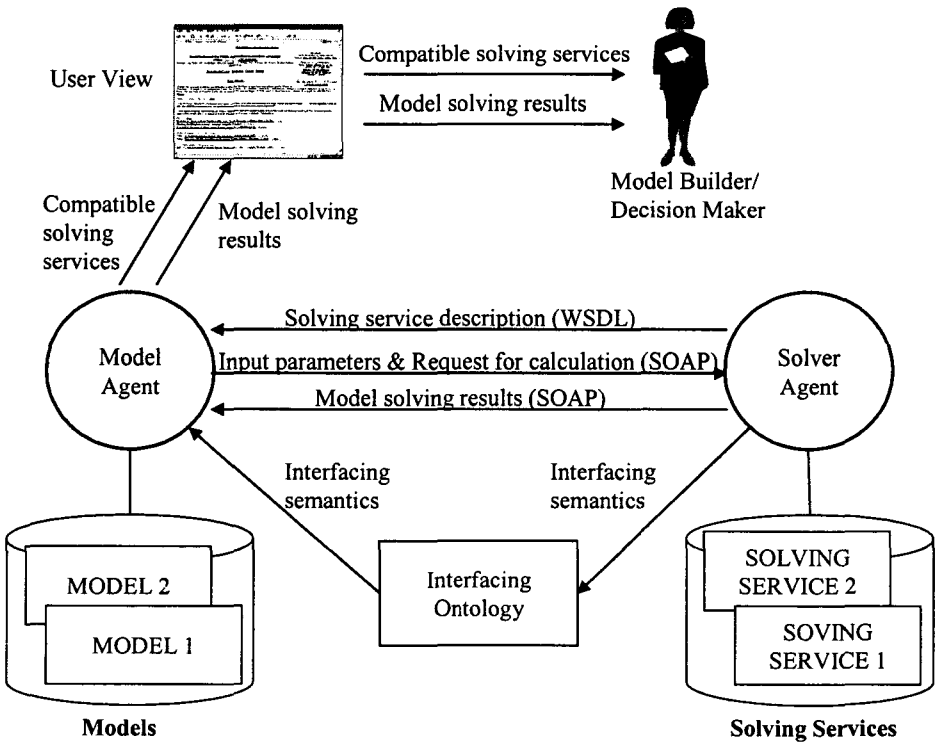

Figure 1 - Conceptual Architecture of the Web Services Based MMS.

represent the following two types of information: the compatibility of a solving service for individual models and their parameter matching patterns.

First, for representation of the model-solver compatibility, models are classified into groups (i.e., a model taxonomy) in such a way that every model in a group can share solvers with one another. By assigning solving services to the individual model groups, the interfacing ontology can represent which models a solving service can be applied to. Figure 2 shows an example taxonomy of optimization models and assignment of compatible solvers. In model management literature, such model taxonomies have been proposed implicitly or explicitly to organize similar models into groups. However, since those taxonomies are usually concerned only with structural assumptions within the models such as whether a parameter of a model is continuous or discrete, we use more detailed taxonomies classifying models one step further in terms of the shareability of solvers.

Second, for representation of the parameter matching patterns between individual models and their compatible solving services, a parameter mapping dictionary (PMD) is constructed. The PMD manages all the possible types of mapping relationships between model parameters and solver parameters in a tabular form where each row represents an individual parameter mapping (Table 1). Typically, every row of the dictionary has the six attributes (i.e., columns): *model*, *solver*, *model parameter(s)*, *solver parameter(s)*, *mapping type*, and *transformation function*. The attributes *model* and *solver* indicate the model and solving service names that the parameter mapping is applied to; the attributes *model parameter(s)* and *solver parameter(s)* indicate the corresponding parameter names of the model and solving service to be linked; the attribute *mapping type* determines whether it is for an input parameter or an output parameter; finally, the attribute *transformation function* is for an algebraic formula to be applied to the parameter value of the sending parameter to produce the data value required in the receiving parameter.

Specifically, *transformation function* is built on two kinds of operands, parameter references and environmental variables. A parameter reference means a parameter value and is denoted by a parameter name enclosed by ampersands (&) at both ends (e.g., &PARAM 1&); an environmental variable refers to a pre-defined system variable and is denoted by a capital string (e.g., TODAY for the current date). These operands can be recursively composed by a set of operators. The operators include domain casting operators for converting the data domain of a parameter (e.g., TO_NUM for conversion to integer and TO_STR for conversion to string), aggregation operators for converting the data cardinality of a parameter (e.g., SUM for summation of a parameter list and AVG for average of a parameter list), and other mathematical operators for various numerical calculations (e.g., LOG for logarithm and SQRT for square root).

Meanwhile, in specifying the port mappings in the PMD, we can distinguish them according to the cardinalities, i.e., 1:1, 1:*n*, *n*:1, or *n*:*m*. An 1:1 or *n*:1 mapping is specified in a single row of the dictionary; an 1:*n* mapping is converted to

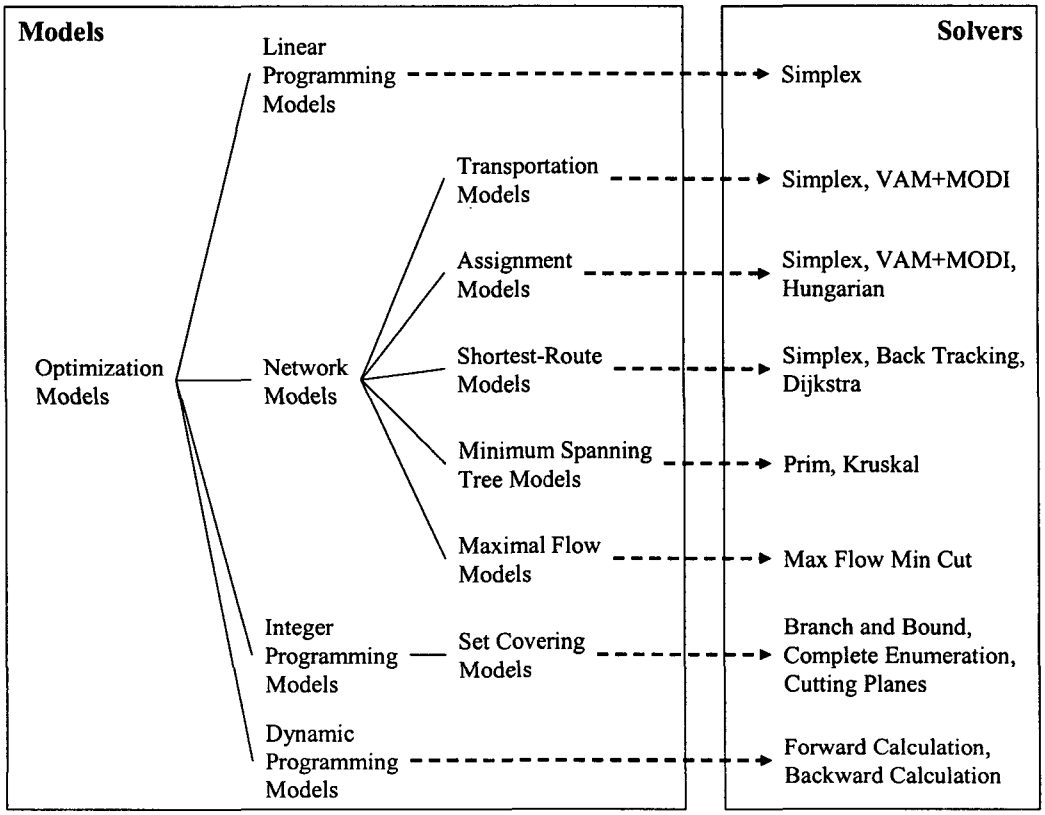

Figure 2 - An Example Taxonomy of Optimization Models and Compatible Solver Assignment

n 1:1 mappings and thus specified in *n* rows; an *n:m* mapping is converted to *m n*:1 mappings and thus specified in *m* rows. For example, in Table 1, the first row indicates the 1:1 mapping between PARAM 1 and PARAM 6, and the second row describes the *n*:1 mapping between the three parameters (PARAM 2, PARAM 3, and PARAM 4) and PARAM 7 by averaging the three parameters. The 1:*n* mapping between PARAM 5 and the two parameters (PARAM 8 and PARAM 9), shown in the third and fourth rows, is converted to two 1:1 mappings.

Model and Solver Agents

On the basis of the interfacing ontology, two software agents, model agent and solver agent, are defined at a higher level to perform the autonomous solver suggestion and intelligent model solution. Referring to the ontology, these two agents cooperate with each other for exchanging parameter values between models and solving services.

The model agent acts as an intermediary between a user view and the solver agent to support a user in solving a model. Specifically, the model agent brings two types of information to the user view: compatible solving services with the model and the model solving results. First, when a user view references a model, the model agent consults the interfacing ontology and identifies the compatible solving services. Then, via the user view, the model agent suggests them for the user to select one. Second, when the user selects a solving service to be used, the model agent requests for description of the selected solving service from the solver agent. According to the description, the model agent sends a Simple Object Access Protocol (SOAP) [13] message containing the input parameters required in the solving service and the request for execution of model-solving calculation. In creating the message, the model agent refers to the interfacing ontology again to understand the parameter matching patterns between the model and the solving service. Afterwards, the model agent delivers the solution returned by the solver agent to the user view.

In this context, the solver agent has the following two responsibilities in response to the model agent's request: sending the service description of a solving service and executing the model-solving calculation. First, the service description defines the solving service's interface such as the message formats, data types, and transport protocols that should be used in the service. Such service description is written in a standardized description language such as Web Service Description Language (WSDL) [13]. Second, when the solver agent receives the request for model-solving calculation from the model agent, it executes the solving service and returns the results to the model agent in a SOAP message format.

Conclusions

A MMS can achieve high productivity and effectiveness in supporting a user's model-solving activities by facilitating the solver reuse satisfying the following two functional requirements: autonomous solver suggestion and intelligent model solution. Having recognized these requirements, this paper proposes an ontology design managing the compatibility and the parameter matching patterns between individual models and their solvers. In designing the ontology, we use the Web services technologies to integrate the models and the solvers in unified system architecture. Also, we define the model and solver agents, which perform the solver suggestion and model solution processes by cooperating with each other based on the model-solver compatibility and the parameter matching patterns provided by the interfacing ontology.

In future research, the intent is to focus on elaborating the interfacing ontology to provide more concrete and formal specifications. A simple ontology that has a form of mapping table has been constructed but it is being extended based on the Resource Description Framework (RDF) [13]. Also, a prototype MMS incorporating the interfacing ontology design is being developed with JAVA programming language.

References

[1] Beynon, M., Rasmequan, S., and Russ, S. (2002). "A New Paradigm for Computer-Based Decision Support," *Decision Support System*, Vol. 33, pp. 127-142.

[2] Bhargava, H., Krishnan, R., Roehrig, S., Casey, M., Kaplan, D., and Müller, R. (1997). "Model Management in Electronic Markets for Decision Technologies: A Software Agent Approach," *Proceedings of the 30th Hawaii International*

Table 1 - A Conceptual Structure of the Parameter Mapping Dictionary

Model	Solving Service	Model Parameter(s)	Solver Parameter(s)	Mapping Type	Transformation Function
MODEL 1	SERVICE 1	PARAM 1	PARAM 6	Input	
MODEL 1	SERVICE 1	PARAM 2, PARAM 3, PARAM 4	PARAM 7	Input	&PARAM 7& = AVG(&PARAM 2&, &PARAM 3&, PARAM 4&)
MODEL 1	SERVICE 1	PARAM 5	PARAM 8	Input	&PARAM 8& = &PARAM 5& / 10
MODEL 1	SERVICE 1	PARAM 5	PARAM 9	Input	&PARAM 9& = &PARAM 5& / 100

Conference on System Sciences, pp. 405-415.

[3] Bolloju, N., Khalifa, M., and Turban, E. (2002). "Integrating Knowledge Management into Enterprise Environments for the Next Generation Decision Support," *Decision Support Systems*, Vol. 33, pp. 163-176.

[4] Brooke, A., Kendrick, D., Meeraus, A., and Raman, R. (1998). "GAMS: A User's Guide," GAMS Development Corporation, http://www.gams.com/docs/gams/GAMSUsersGuide.pdf.

[5] Dolk, R.D. (2000). "Integrated Model Management in the Data Warehouse Era," *European Journal of Operational Research*, Vol. 122, pp. 199-218.

[6] Dutta, A. (1996). "Integrating AI and Optimization for Decision Support: A Survey," *Decision Support Systems*, Vol. 18, pp. 217-226.

[7] Fourer, F., Gay, D.M., and Kernighan, B.W. (1990). "A Modeling Language for Mathematical Programming," *Management Science*, Vol. 36, pp. 519-554.

[8] Geoffrion, A.M. (1989). "The Formal Aspects of Structured Modeling," *Operations Research*, Vol. 37, pp. 30-51.

[9] Gregg, D., Goul, M., and Philippakis, A. (2002). "Distributing Decision Support Systems on the WWW: the Verification of a DSS Metadata Model," *Decision Support Systems*, Vol. 32, pp. 233-245.

[10] Huh, S. (1993). "Modelbase Construction with Object-Oriented Constructs," *Decision Science*, Vol. 24, pp. 409-434.

[11] Rizzoli, A.E., Davis, J.R., and Abel, D.J. (1998). "Model and Data Integration and Re-use in Environmental Decision Support Systems," *Decision Support Systems*, Vol. 24, pp. 127-144.

[12] Shim, J.P., Warkentin, M., Courtney, J.F., Power, D.J., Sharda, R., and Carlsson, C. (2002). "Past, Present, and Future of Decision Support Technology," *Decision Support Systems*, Vol. 33, pp. 111-126.

[13] World Wide Web Consortium. (2003). http://www.w3.org/.