International Journal of Management Science
Vol 5, No 1, May 1999

WebIME: An Web-based Integrated Modeling Environment for Multi-facetted Model Representation and Management

HyoungDo Kim
EC-Internet Development Team, SungJi Building,
3Ga, HanGangRo, YongSanGu, Seoul, Korea

JongWoo Kim
Department of Statistics, ChungNam National University
Yusong-gu, Taejon, 305-764, Korea

SungJoo Park
Graduate School of Management, KAIST
Kusong-dong, Yusong-gu, Taejon, 305-701, Korea

(Received October 1998 ; revision received February 1999)

ABSTRACT

WebIME is an Web-based integrated modeling environment that implements a multi-facetted modeling approach to mathematical model representation and management. Key features of WebIME include the following: (i) sharing of modeling knowledge on the Web, (ii) a user-friendly interface for creating, maintaining, and solving models, (iii) independent management of mathematical models from conceptual models, (iv) object-oriented conceptual blackboard concept, (v) multi-facetted mathematical modeling, and (vi) declarative representation of mathematical knowledge. This paper presents details of design and implementation issues that were encountered in the development of WebIME.

1. INTRODUCTION

WebIME (Web-based Integrated Modeling Environment) is to provide organizations the ability to create, store, and solve models for decision making on the Web. To implement such a modeling environment (ME) [16, 37] or a model/ modelbase management system (MMS) [1, 3, 22, 23, 29, 35], a conceptual modeling framework is required for representing and managing mathematical models. Some dis-

tinctive frameworks for the purpose are structured modeling [14], logic-based modeling [1, 28], graph grammar [23, 24], object-oriented modeling [22, 35], and frame-based modeling [2, 30, 31]. The researches advance the formality, generality, executability, and implementability of the frameworks.

As the bases of integrated modeling environments, however, current modeling frameworks have limitations: lack of facility to coordinate different users' perspectives and lack of mechanism to manage modeling knowledge integratively. Modelers need various different kinds of knowledge including the specification of objects, their relationships, and properties contained in a problem domain, assumptions about mathematical relationships, and model management. To share the knowledge among various users, an ME needs to be based on a modeling framework, which supports different abstraction points of view among the users and assumption management.

WebIME is based on the core concepts of a multi-facetted model representation and management framework for supporting the knowledge sharing and management. Refer to Kim et al. [26, 27, 36, 38, 39] for details. WebIME provides the following attractive features to users: (1) sharing of modeling knowledge on the Web; (2) a user-friendly interface for creating, maintaining, and solving models; (3) independent management of mathematical models from conceptual models; (4) object-oriented conceptual blackboard concept; (5) multi-facetted mathematical modeling; and (6) declarative representation of mathematical knowledge.

There are several graphical and non-graphical modeling approaches, languages, and systems. Non-graphical ones lack user-friendly features such as model construction using graphical inputs. Graphical ones, with few exceptions such as GBMS/SM[7] and BLOOMS[13], are either domain-specific or support only some stages of a model's life cycle. However, all of them assume stand-alone environment for model building. With WebIME, a group of distributed users, such as modelers and decision makers, are able to compose and share models and their mathematical assumptions on the Web. A graphical tool is provided for mathematical modeling in addition to syntax-directed editors for conceptual and mathematical modeling. This paper describes in detail how such attractive features are realized in WebIME.

The remainder of this paper is organized as follows. Section 2 presents the basic principles underlying the WebIME. Section 3 explains WebIME from the perspective of an end user. The implementation architecture of WebIME is presented in Section 4. Finally, contributions of this paper and further research directions are summarized in Section 5. Appendix A contains an example problem for explaining distinctive characteristics of WebIME.

2. BASIC PRINCIPLES OF THE APPROACH

WebIME have two fundamental objectives, i.e. to reduce the complexities involved in creating and manipulating models, and to support organizational members to share modeling knowledge and to cooperate. WebIME is based on the multi-facetted modeling framework that employs some basic principles. They are as follows.

2.1 Model-Data Independence

Model data independence refers to the specification of a model schema that is independent of the data used for generating a model instance. This kind of independence promotes the use of a single model schema for the generation of multiple model instances.

2.2 Model-Solver Independence

Model-solver independence pertains to a model representation scheme that is solver-neutral and its expressive power unconstrained by any solver. This facilitates the use of multiple sovlers in a modeling environment.

2.3 Independence of Mathematical Models from Conceptual Knowledge

Structural knowledge identifies real or conceptual objects, relationships, and properties in a problem domain. To enhance the reusability and sharing of conceptual knowledge, it is managed independently of mathematical models in the multi-facetted approach. We call the structural knowledge set "object-oriented conceptual blackboard" because various modelers sharing the knowledge use consistent terms in communicating about the problem domain and find some mathematical models already defined using the knowledge and related assumptions. A conceptual blackboard in a problem domain is the cue to model and manage mathematical assumptions.

2.4 Multi-facetted Concept

Mathematical modeling can be defined as identifying related components in a problem domain and making mathematical assumptions [44]. In the multi-facetted approach, facet concept is proposed to systematically manage mathematical assumptions of various modelers within an environment, normally used in connection with structural knowledge. A facet is defined as an aspect of a property of an object or relationship. The facet concept accommodates different points of view about a property and allows modelers flexible use of properties in their mathematical models. For example, production cost of a product can be estimated

using its material and labor costs when its transportation cost is small compared with its total production cost. But if the transportation cost is on the rise and can not be disregarded, the estimation rule should take account of the transportation cost. As the example shows, the set of properties to determine another property's value or mathematical relationships among the properties can be defined differently according to modeling purpose, abstraction level, and/or modeler's point of view. Furthermore, a property can be used as an input item in one case and an output item in another case. In the above example, the production cost is an output item, but can be an input item in another model for estimating its total cost.

2.5 Declarative Representation of Mathematical Knowledge

Most modeling languages such as SML [14, 17, 18], GAMS [6] and AMPL [12] support constructs for defining artificial index sets, which are good for representing mathematical knowledge algebraically in compact form [19, 41]. But it is hard for ordinary users to define or understand complex models composed of many index sets. Even for experts, understandability and readability degrade as the number of index sets increases.

In the multi-facetted approach, only the names of objects are allowed as index sets. We tried to improve comprehensibility and consistency of modeling knowledge by prohibiting modelers from defining artificial index sets. Along the line of improvement, declarative representation of mathematical relationships are supported for modelers who are poor at mathematical representation.

2.6 Open Systems Architecture

Open systems architecture permits a modeling system to support various external systems such as solvers and databases. As a collaboration platform, WebIME employs the Web which demonstrates how open systems architecture can support sharing information among large dispersed groups on the Internet and intranets.

3. USER'S VIEW OF WEBIME

This section describes the WebIME prototype with the aid of the multi-commodity transportation example [12] presented in Appendix A. A multi-facetted representation of the problem is completely listed in Kim et al. [27].

According to the approach, mathematical modeling and analysis can be performed by following the procedure presented in figure 1. Modelers usually describe a target problem informally in the initial phase of the problem analysis.

Such a problem description includes a mixture of problem domain, components, assumptions, constraints, and objectives. Object modeling phase is to abstract conceptual objects contained in a target problem and their relationships. Referring to the object-oriented conceptual blackboard that contains something already defined and managed, a modeler retrieves objects, relationships, and properties related with the problem domain and adds some new ones if needed. According to the refined definition, model developers just add or update instances of objects and relationships considered to be captured for the problem solving. In the third phase, mathematical assumptions among the properties retrieved or added in the previous phase are defined using the facet concept. Model developers reuse some mathematical assumptions if they are available. New assumptions could be constructed through revision of existing ones. Those assumptions are externally represented as a facet graph, which is a kind of acyclic graph. The fourth phase is to define how to generate a model instance from the facet graph defined in the previous phase and the conceptual blackboard. The definition includes how to gener ate a model instance and how to solve the model instance with a solver. In the phase of model execution, a model instance in a form that can be processed using a solver is created within the modeling environment, which executes it with the solver. The model instance is automatically created from the model instance database with its model instance definition. The execution results are presented to users for model analysis or reflected into the model instance database.

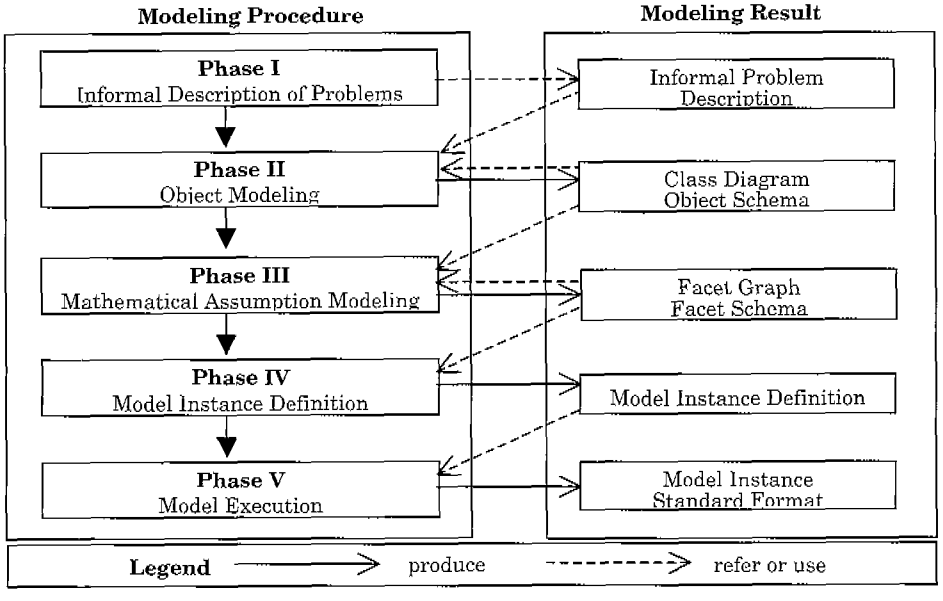

Figure 1. Modeling Procedure in the Multi-facetted Approach

3.1 Object Modeling

Object modeling starts with identifying conceptual objects, relationships and properties needed for modeling a given problem. New ones are added to the conceptual blackboard or ones already existing in the blackboard are referred. The blackboard can be externally represented by any object-oriented notations including UML [4] and object-relationship (OR) diagrams [27]. Currently, we are using simplified class diagrams to provide an abstract viewpoint of structural knowledge to modelers. Components of a simplified class diagram are objects, isA relationships, isAggregateOf relationships, and user-defined relationships. Objects are represented by rectangles and user-defined relationships by diamonds. The class diagram of the example problem is figure 2, where a bounded rectangle means that all the objects and relationships within the rectangle have isAggregateOf relationship with the bounded object. That is, trans_system is the aggregate of all the objects within the rectangle. Detailed object and relationship information is described in object schemata.

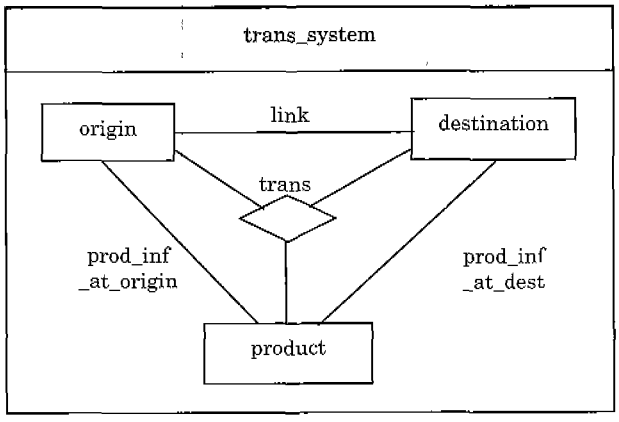

Figure 2. Class Diagram for the Multi-commodity Transportation Example

Figure 3 shows the hierarchy of objects contained in the conceptual blackboard, which plays the role of main anchor to retrieve needed objects from the blackboard. The hierarchy consists of objects for the mult-commodity transportation example as well as ones for system management such as DataType and Relationship.

Each line of the hierarchy contains two kinds of hyperlinks to the object schema and the list of object instances belonging to the schema. When a modeler clicks on an object name, its hyperlink guides him/her to the page of its schema. For example, figure 4 shows the schema of object 'Link'. When (s)he clicks on tri-

ple asterisks (***) in a line of the hierarchy, its hyperlink guides him/her to the list page of its object instances. For example, figure 5 shows the list of 'Link' instances.

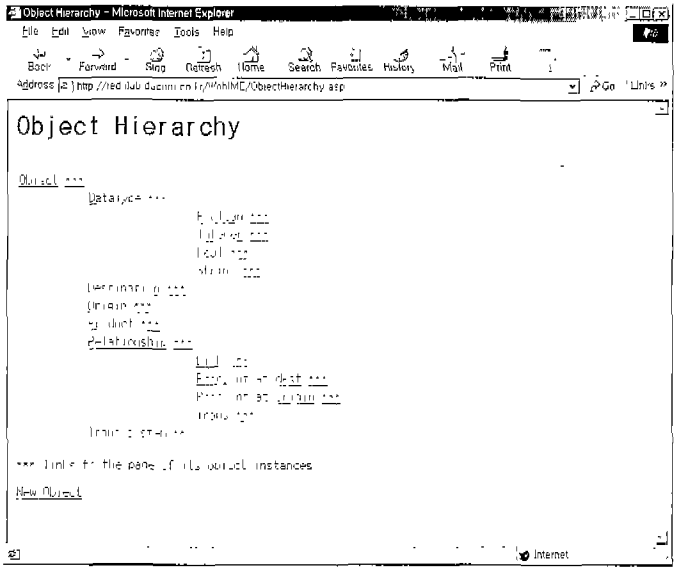

Figure 3. Object Hierarchy

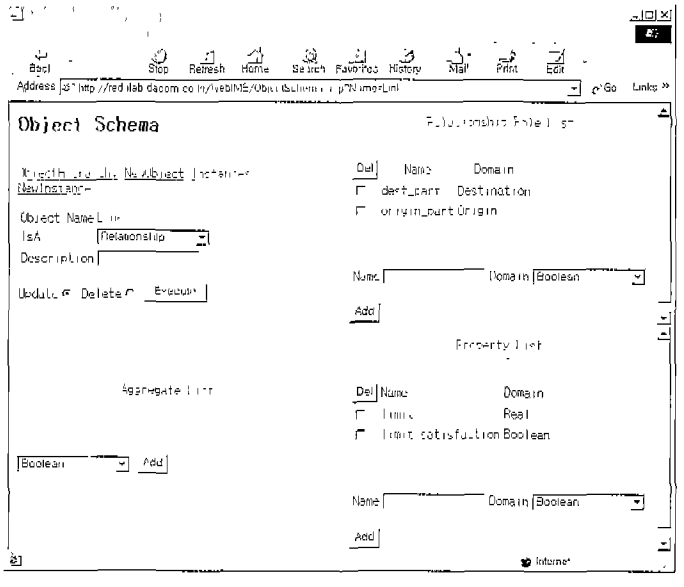

Figure 4. Object Schema

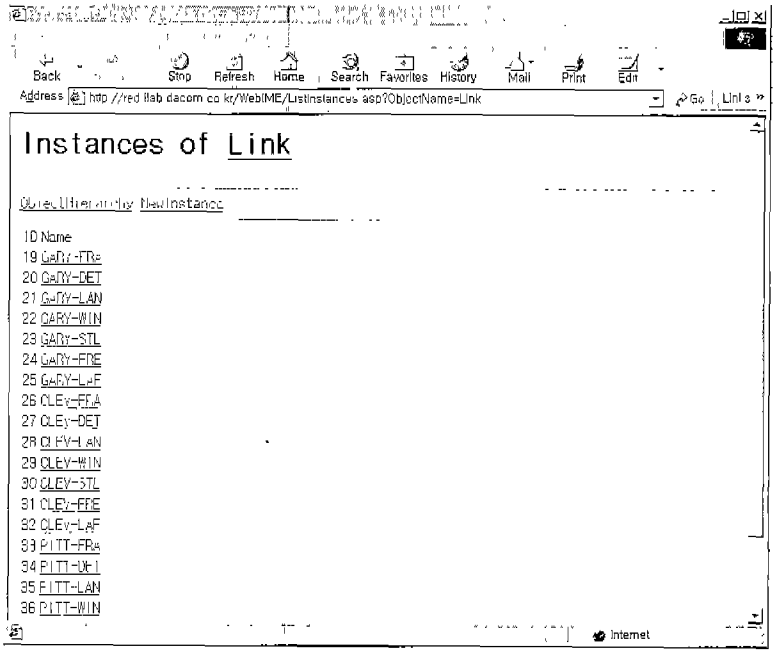

Figure 5. List of Object Instances

An object schema is composed of four parts: object idenfication, aggregate specification, relationships with other objects and relationships, and properties. The first part of an object includes its name, textual description, and its parent object described by an IsA relationship. The second part can be omitted if it is not an aggregate. An object can become an aggregate component by clicking the 'Add' button. Such a component has minimum and maximum numbers of participation in the aggregate. The third part only applies to user-defined relationships and specifies participating objects and their roles. The fourth part defines properties with their names and domains. For example, in figure 4, object "link" has properties "limit" and "limit_satisfaction" whose domains are "Real" and "Boolean", respectively.

WebIME supports two kinds of object instance modeling: syntax-directed editing and mapping from external databases. Syntax-directed editing of a 'Link' instance is given in figure 6, where relationship role and property values can be specified. Figure 7 shows the unique instance of the object 'Trans_system', which shows instances of its aggregate components.

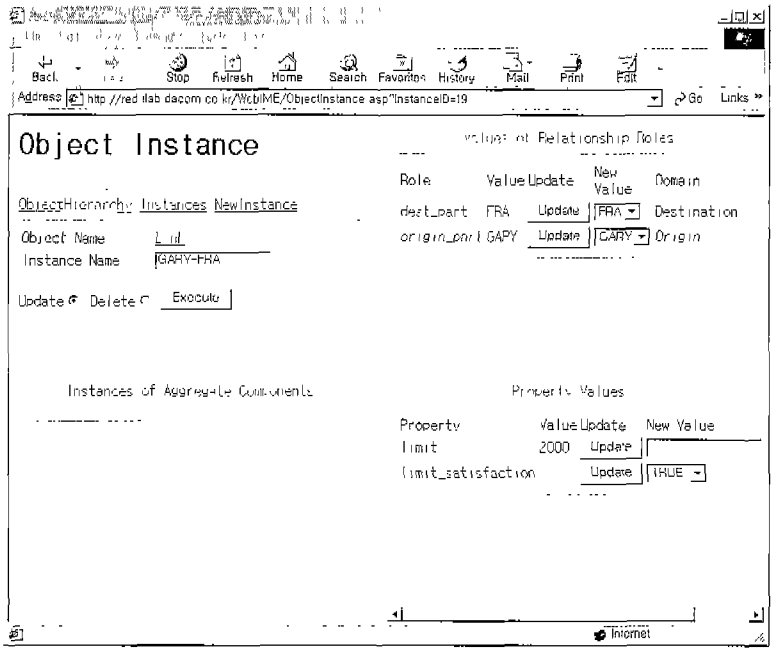

Figure 6. Syntax-directed Editing of an Object Instance

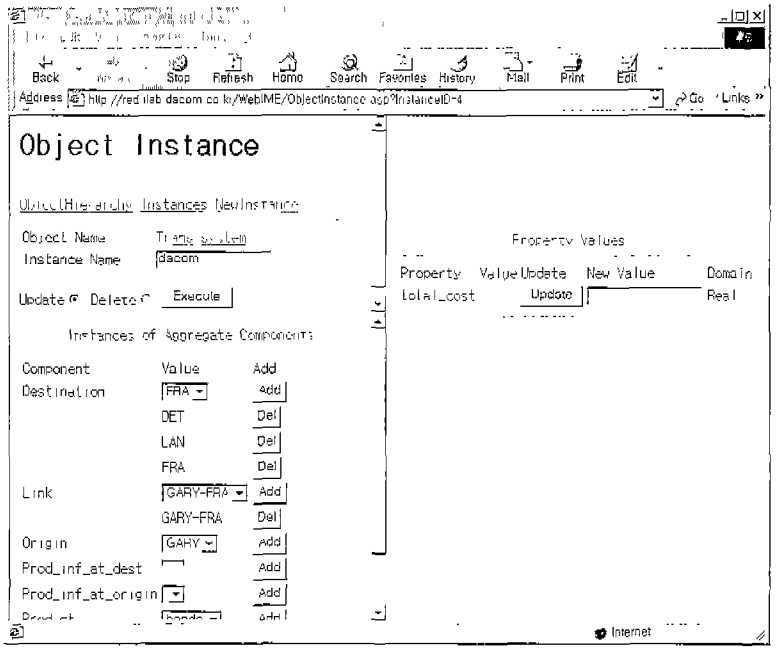

Figure 7. The Unique Instance of 'Trans_system'

3.2 Modeling of Mathematical Assumptions

Modeling of mathematical assumptions starts with identifying facets related with a given problem. New facets are added to the model to be made or ones already existing in the other models are referred. A facet graph is an acyclic graph specifying facets and their definitional dependencies [14]. In the graph, directed arcs represent definitional dependencies, where its head node is called a "calling facet" and its tail node a "called facet".

When F is defined as a facet of a property P of an object O, O and P is called "origin object" and "origin property" respectively and we say that "facet F originates from the property P of the object O." A facet is either a primary facet or a derived facet. The facet located in a root node of a facet graph is a primary facet. All the other nodes are derived facets. Details about each facet is described in a facet schema, which include facet name, originating object and property, and facet type (identifying whether it is a primary or derived one). In the case of derived facets, called facets and computational/comparison rules should be additionally described.

WebIME supports two modes of mathematical modeling: syntax-directed editing and graphical modeling. Model components (i.e., model and facet schemata) can be systematically described using HTML forms in the first mode. Figure 8 shows an anchor page for model schemata, facet schemata, and model instance definitions. A model schema includes its name, description, and component facet schemata as figure 9 demonstrates. Note that modelers can specify or change graphical characteristics of component facet schemata such as location and size. Component facet schemata describe the details of mathematical assumptions. Figure 10 shows a schema of facet 'limit_satisfaction_test' derived from 'limit_satisfaction' property of 'Link'. Note that it has a facet rule 'LE' for comparing callees: 'limit_p' and 'trans_qty_p'.

WebIME provides a friendly graphical user interface containing tool bars, menus and dialog boxes for synthesizing models. Modelers can create or review facet graphs using the graphical modeling tool. Figure 11 is a screen shot of editing a facet graph. By clicking on the 'F' icon of the tool bar, they can create a facet, whose schema is described using dialog boxes containing forms similar to the syntax-directed editing. By clicking on the arrow icon of the tool bar, they can create a facet graph edge, that is, definitinal dependency between two facets. Other icons can be used for the decoration of facet graphs.

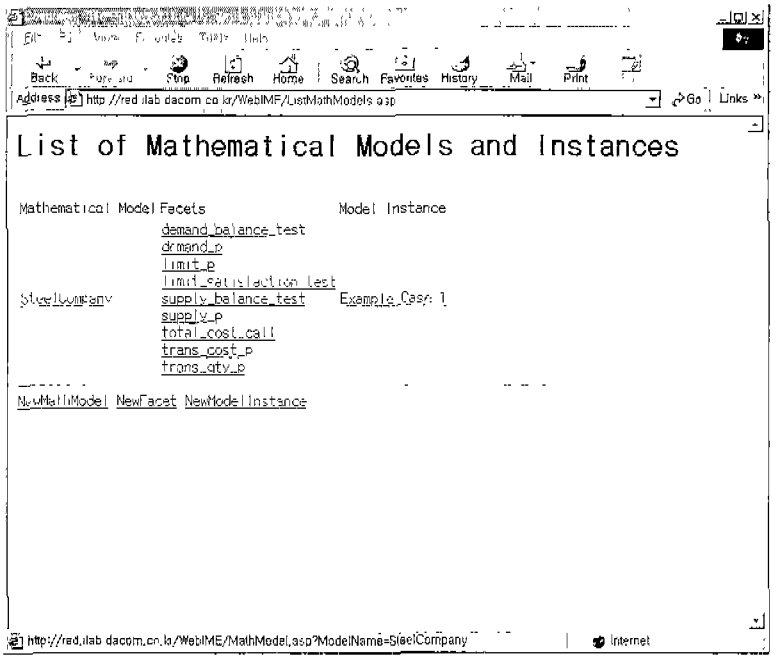

Figure 8. Lists of Mathematical Models and Instances

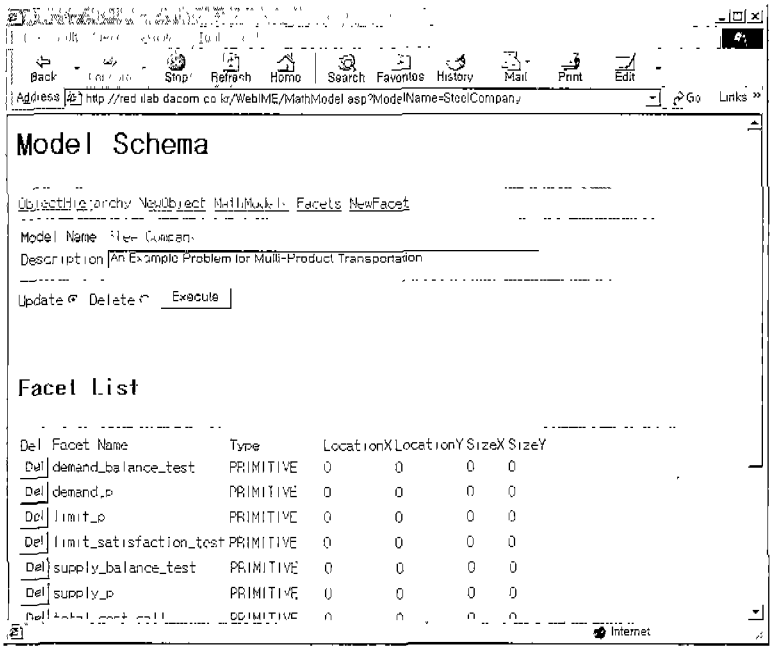

Figure 9. Model Schema

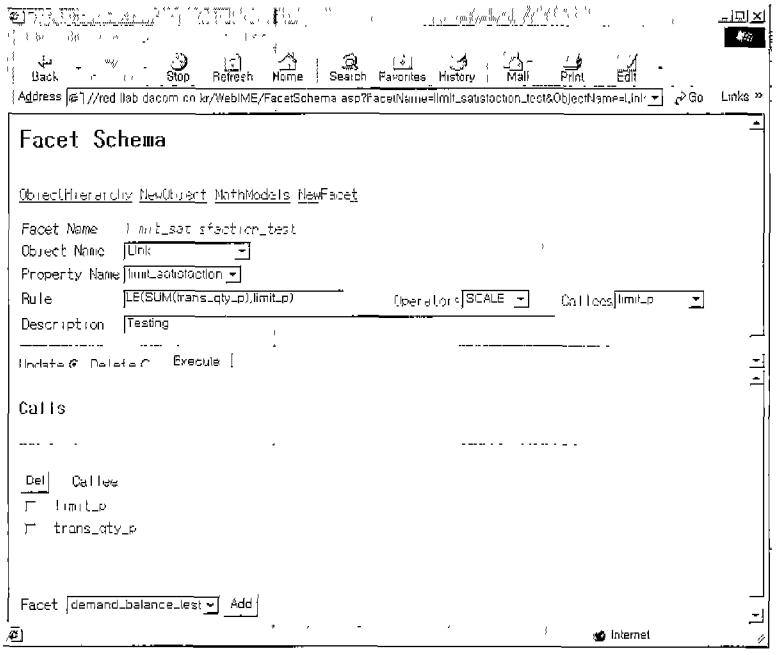

Figure 10. Facet Schema

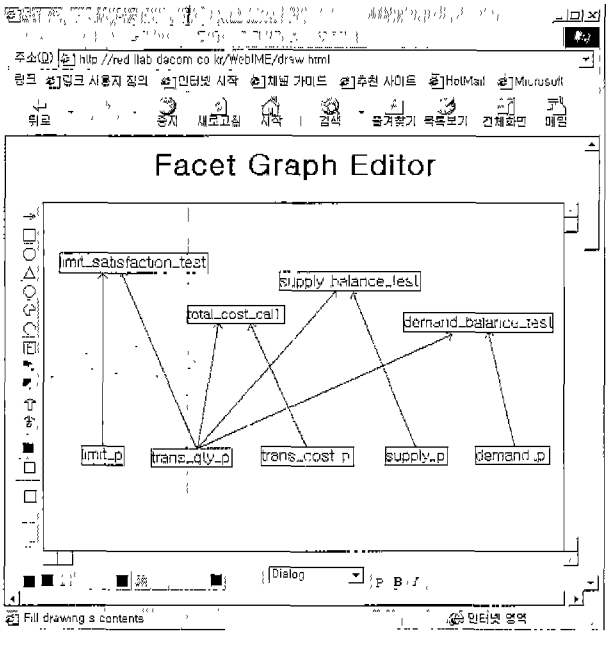

Figure 11. Facet Graph Editor

3.3 Model Instance Definition and Solution

In the multi-facetted approach, model instance definitions are distinguished from model instances for creating dynamic model instances based on object instances. Accordingly, model instance definitions are useful for real-time modeling environments where object instances change frequently.

A model instance definition specifies how to compose and solve an instance of some model already defined in the previous phases. It defines its name, model name, selection rules and solver description. Object instances participating in a model solution are those of objects from which primitive facets are originating in the facet graph. A selection rule specifies what instances are to be selected for model solution in a form of SQL. If an object has no selection rule, that means all the instances be selected for model solution. Solver description includes solver name, which can be used for getting a solution with the instance data, and other information necessary for the solver execution. For example, facets for objective function, decision variables and constraints, and objective type (min or max) are necessary for LP solver execution.

As in figure 12, a model has to be selected among the ones enlisted in the box. A dialog box asks a modeler for specifying input items required for executing a solver when (s)he clicks on the box to select it from a list of solvers. Up to now, WebIME supports two different kinds of solvers: the internal evalator and an LP solver. SQL-like selection rules as in 4GLs (4^{th} generation languages) such as PowerBuilder™ [40] can be added or deleted interactively.

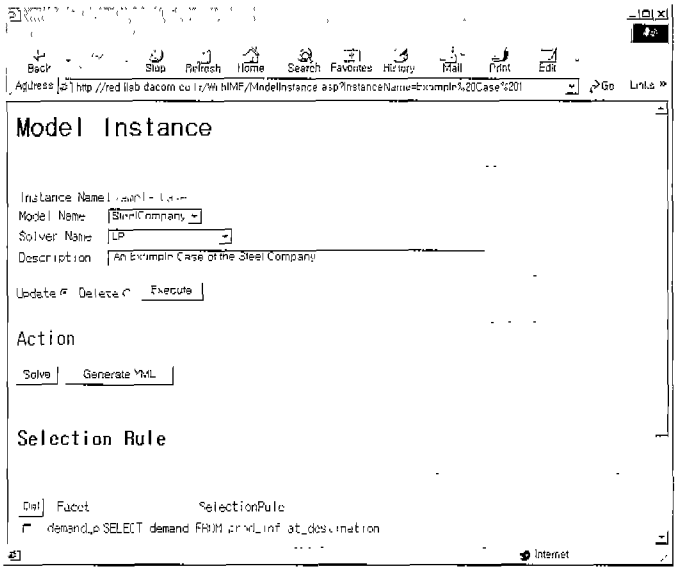

Figure 12. Model Instance Definition

To refer to the results of a decision model, a decision maker or his(her) supporters select a model instance definition and direct WebIME to compose and execute a model instance with a solver. Model solution commences by clicking the "Solve" button in the model instance definitions. WebIME has been designed to be able to add a new solver to it by registering its name along with "Solver Input Format Specification" and "Solver Output Handling Specification" [8].

XML (Extensible Markup Language) [5] specification of a model and its instance is a fundamental solution for trading and/or solving them on another machine. WebIME generates such a specification when you click on the XML button in figure 12. Any solvers that support the specification will have the ability to communicate models and solutions with customers. The markup language for model representation and management on the Web is based on a conceptual modeling framework, called Object-Oriented Structured Modeling (OOSM), which is an extension of the structured modeling. For details, refer to Kim [25].

4. WEBIME IMPLEMENTATION ARCHITECTURE

The implementation architecture underlying WebIME is illustrated in Figure 13. There are various sub-systems shown in the figure that can be classified by the following categories: conceptual modeling, mathematical modeling, storage and retrieval, and solution.

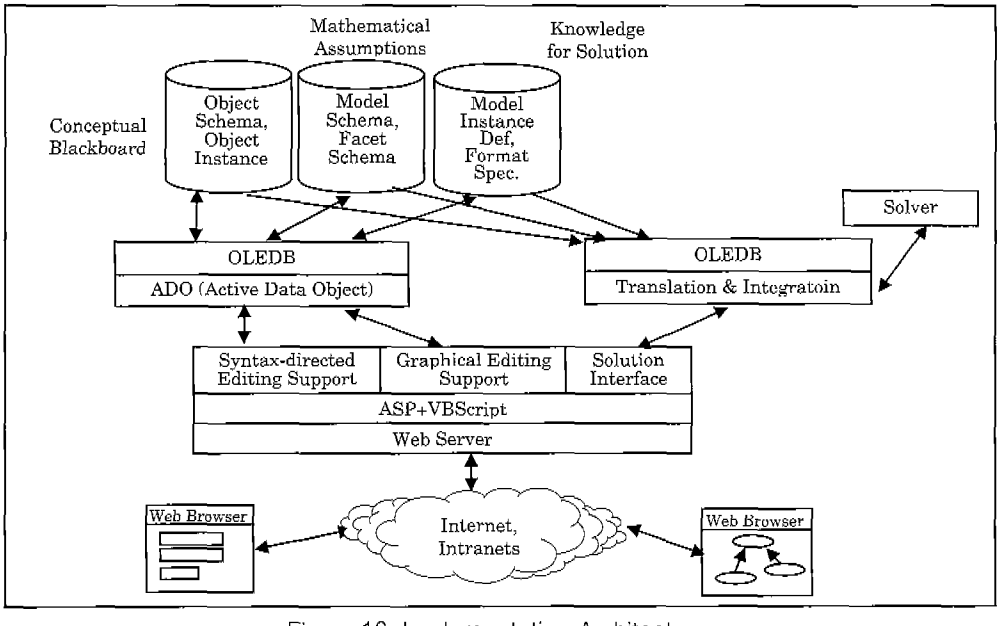

Figure 13. Implementation Architecture

4.1 Conceptual Modeling

This sub-system includes syntax-directed editors for modeling object schemas and instances. They are implemented using HTML forms, frames, and scripts that are dynamically generated by server-based scripts such as ASP [21]. In composing those editors, some basic principles are applied. As a basis of modeling, first of all, standard Web technologies are only employed in the client side. Second, the independence of conceptual model and data is strictly observed. Third, WebIME does not allow users to make mistakes or omit steps of a task by dynamic page generation and syntax-directed editing, as a proactive modeling system.

4.2 Mathematical Modeling

This sub-system includes syntax-directed model editors and a graphical modeling tool. As in the conceptual modeling, all the mathematical modeling work can be done with the syntax-directed model editors that are dynamically generated using server-based scripts. The graphical modeling tool is complementary for the editors, that is implemented as a Java Applet [11].

In describing computational rules, artificial index sets are not allowed. Object names are defined as common index sets applicable to any expressions for representing mathematical knowledge. This is to enhance the comprehensibility and readability of computational rules. If a facet originates from a normal object, its index set is the object name. In figure 11, for example, "supply_balance_test" originates from a normal object "origin", so its index set is origin. If a facet originates from a user-defined relationship, its index set is the Cartesian product of the objects participating in the relationship. In figure 11, for example, trans_cost_p originates from "trans" which is a user-defined relationship among objects "origin", "destination" and "product", so its index set is origin × destination × product. If an object participate in a user-defined relationship more than once, role names are used for distinction. Any facet originating from a relationship "manage" between employees, for example, have the index set boss(employee) × worker(employee) where "boss" and "worker" are the role names in the relationship.

The principle for index sets does not assume that a model developer should know all the object names, i.e. index sets, in advance. When a model developer adds an object to a modeling environment, an index set is also added into the environment. A model developer can define the computational rule of a facet if s(he) knows what the originating objects of the facet and its called facets are.

Computational rules are represented as a composite function of basic operators [26]. In the multi-commodity transportation problem, for instance, declarative representation of total transportation cost (2) is a substitute for its algebraic

representation (1). The index sets of "trans_cost_p" and "trans_qty_p" are origin× destination× product and two basic operators '*' and 'SUM ... FOR' are applied in sequence.

$$\sum_{for\ all\ i,j,k} c_{ijk} \times q_{ijk}$$
$$where\ i = origin, j = destination, k = product, \quad (1)$$
$$c_{ijk} = trans\_cost\_p, and\ q_{ijk} = trans\_qty\_p$$

SUM trans_cost_p*trans_qty_p FOR origin, destination, product (2)

4.3 Model Storage and Retrieval

Modules in this sub-system include interface modules for model storage and retrieval. WebIME stores all the model components such as object schemata, object hierarchy, object instances, model schemata, and model instance definitions in relational databases. When detailed information about a new or changed component is notified to a server-based script, it opens a connection to a database through OLEDB [34] objects and stores the information. All the anchors and selection boxes in HTML pages are also generated by server-based scripts retrieving related components from the databases.

4.4 Model Solution

The model execution procedure of WebIME is as follows:

[STEP 1] Create a model instance by retrieving object instances needed for model execution using SQL-type selection rules.
[STEP 2] Select a solver based on the solver description and verify whether the solver can give a solution to the model instance.
[STEP 3] Convert the model instance into a form that can be processed by the solver.
[STEP 4] Execute the model instance with the solver.
[STEP 5] Output the execution results to users and update object instances if needed.

Figure 14 conceptually depicts the extensible approach for solver management in WebIME. To enable the extension, a specification language for "Model Instance Standard Format", "Solver Input Format Specification" and "Solver Output Handling Specification" has been designed [8]. The "Model Instance Stan-

dard Format" is independent of any solver and created in the first phase of the 4 phases for model execution. A model instance that can be processed by a solver is created by using a model instance definition according to a "Model Instance Standard Format" and the "Solver Input Format Specification" of the solver. The result of model execution is reflected in object instances by referencing the "Solver Output Handling Specification" of the solver.

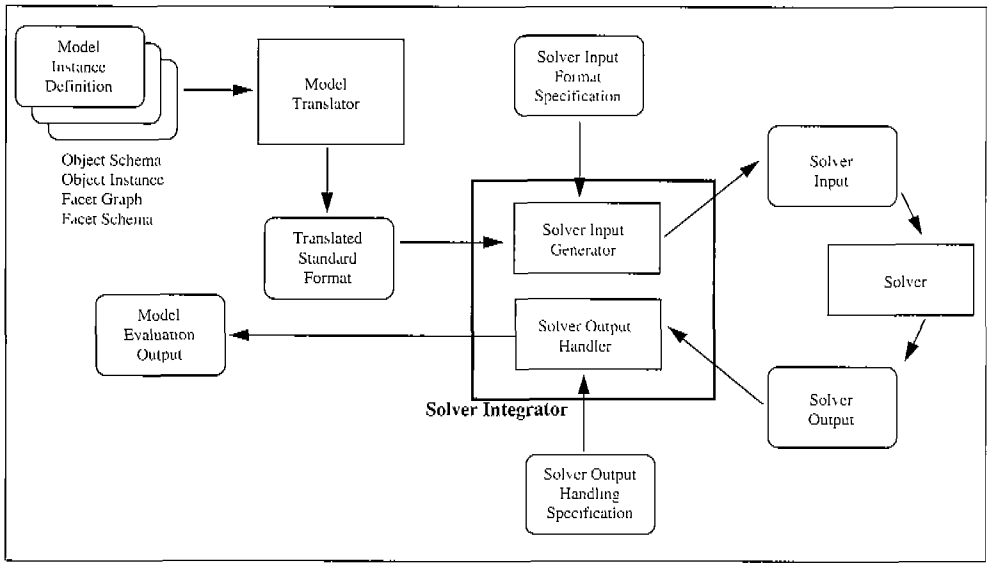

Figure 14. Extensible Solver Management

4.5 Model Sharing

Traditional modeling frameworks and environments were generally developed for single user, but modeling knowledge should be managed in organizational dimension to support collaborative modeling and reuse [35,38,42,43]. For model sharing, it is necessary to support various levels of knowledge abstraction, incorporating different points of view, and problem-centered knowledge representation. Accordingly, multi-facetted modeling approach contributes to model sharing by separating structured knowledge from mathematical knowledge, multi-facetted concepts, and the declarative representation of mathematical knowledge. On the level of mathematical modeling, especially, a part of an existing model can be shared by creating some new facets and composing a model from the existing and newly created facets. The new facets reflect the assumptions that the existing model can not satisfy. This way, sharing of mathematical models by multiple users is supported by flexible management of facets on the Web.

4.6 Basic Principles Supported by the Architecture

The basic principles are supported by the WebIME architecture. The specification of a model schema is independent of the data used for generating a model instance. Furthermore, the use of facet graphs for representing model schemata provides model data independence since graphical representation are unconstrained by model instance data. WebIME also provides a clean separation between solvers and models. Model schema representations are not constrained by any solvers.

Independent management of conceptual objects and instances from mathematical model schemata and model instance definitions provides many benefits. First, common conceptual background can be shared among model developers using WebIME. Second, multiple facets and models are effectively managed and upgrade the reusability of conceptual objects and instances. Thus, they play the role of a conceptual blackboard.

Only the names of objects are allowed in WebIME as index sets. Comprehensibility and consistency of modeling knowledge are accomplished by prohibiting modelers from defining artificial index sets. WebIME also supports declarative representation of mathematical relationships that is an effort to conceptualize comparison/computation rules. The symbolic representation is then used to infer or constrain other facts with facet information.

Open systems architecture permits WebIME to integrate various external systems such as databases and solvers. As a collaboration platform, WebIME demonstrates how to share model information among large dispersed groups on the Internet and intranets.

5. CONCLUSION

This paper presents a prototype modeling environment called WebIME, which supports multi-facetted model formulation, maintenance, and solutions on the Web. Multi-facetted representation and management of mathematical modeling knowledge are systematically supported for its reuse and sharing among users on the Internet and intranets. WebIME has many tools for conceptual modeling, mathematical modeling, storage and retrieval, and solution that include syntax-directed editors for conceptual objects and instances, graphical editors for facet graphs, syntax-directed editors for mathematical models.

Major contributions of this paper are as follows:

- Although there are several graphical and non-graphical modeling approaches, languages, and systems, they generally assume stand-alone environment for

model building. With WebIME, a group of distributed users, such as modelers and decision makers, are able to compose and share models and their mathematical assumptions on the Web. Multi-facetted modeling approach contributes to the model sharing by separating structured knowledge from mathematical knowledge, multi-facetted concepts, and the declarative representation of mathematical knowledge.

- Hypertext [9] and hypermedia based on the Web technologies are applied to the multi-facetted approach for providing nonlinear links among related concepts, where users can make use of dynamic links among related items of the conceptual blackboard, facet schemata, facet graphs, and model instance definitions.

Further research directions are as follows:

- WebIME adopts declarative modeling of computational rules and does not allow any index sets except object names in specifying the rules. The expressive power of the specification technique has to be evaluated. For some users familiar with algebraic representation, we plan to develop a tool to support automatic conversion between algebraic and declarative representations.
- Model integration has two different meanings in the area of MMS: deep integration and functional integration [20]. The former is to make a new model by integrating more than two models. The new one is represented in the same modeling framework as the component models [15, 32, 33]. The latter is to make a new model that is not represented in the same framework as the component models. It is proposed as a form of model interconnection language or model description language to represent computational sequence, input/output relationship, etc. among models [10, 35]. We are currently doing research for deep integration, where two facet graphs can be integrated into a new one.

REFERENCES

[1] Bhagava, H.K. and S.O. Kimbrough, On Embedded Languages for Model Management, *Decision Support Systems* 10(3)(1993), 277-299.

[2] Binbasioglu, M. and M. Jarke, Domain Specific DSS Tools for Knowledge-based Model Building, *Decision Support Systems* 2(3)(1986), 213-223.

[3] Blanning, R.W., A Relational Framework for Join Implementation in Model Management Systems, *Decision Support Systems* 1(1)(1985), 69-82.

[4] Booch, G., I. Jacobson and J. Rumbaugh, *The Unified Modeling Language for Object-Oriented Development*, Rational Software Corporation, Monterey, CA, 1997.

[5] Bray, T., J. Paoli and C.M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0, *http://www.w3.org/TR/1998/ERC-xml-19980210*, 1998.

[6] Brooke, A.D., D. Kendrick and A. Meeraus, *GAMS: A User's Guide*, Scientific Press, Redwood City, CA, 1988.

[7] Chari, K. and T.K. Sen, An Implementation of a Graph-based Modeling System for Structured Modeling (DBMS/SM), *Decision Support Systems* 22(1998), 103-120.

[8] Cho, K.H., *An Approach to Solver Integration for Modeling Environment*, MS Thesis, KAIST, TaeJon, Republic of Korea, 1994.

[9] Conklin, J., Hypertext: An Introduction and Survey, *Computer*, Sept. (1987), 17-41.

[10] Dolk, D.R. and J.E. Kottemann, Model Integration and a Theory of Models, *Decision Support Systems* 9(1)(1993), 51-63.

[11] Flanagan, D., *Java in a Nutshell*, O'Reilly & Associates, CA, 1996.

[12] Fourer, R., D.M. Gay and B.W. Kernighan, *AMPL : A Modeling Language for Mathematical Programming*, The Scientific Press, 1993.

[13] Gagliardi, M. and C. Spera, BLOOMS: A Prototype Modeling Language with Object-Oriented Features, *Decision Support Systems* 19(1997), 1-21.

[14] Geoffrion, A.M., An Introduction of Structured Modeling, *Management Science* 33(5) (1987), 547-588.

[15] Geoffrion, A.M., Reusing Structured Models via Model Integration, *Proceedings of the 23rd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, 1989, 601-611.

[16] Geoffrion, A.M., Computer-based Modeling Environments, *European Journal of Operational Research* 41(1989), 33-43.

[17] Geoffrion, A.M., SML : A Model Definition Language for Structured Modeling: Level 1 and 2, *Operations Research* 40(1)(1992a), 38-57.

[18] Geoffrion, A.M., SML : A Model Definition Language for Structured Modeling: Level 3 and 4, *Operations Research* 40(1)(1992b), 58-75.

[19] Geoffrion, A.M., Indexing in Modeling Languages for Mathematical Programming, *Management Science* 38(3)(1992c), 325-344.

[20] Geoffrion, A.M., Structured Modeling: Survey and Future Research Directions, *ORSA CSTS Newsletter* 15(1)(1994), 11-19.

[21] Homer, A. et al., *Professional Active Server Pages*, Wrox Press, 1997.

[22] Huh, S.Y, Modelbase Construction with Object-Oriented Constructs, *Decision Sciences* 24(2)(1993), 409-434.

[23] Jones, C.V., An Introduction to Graph-based Modeling Systems, Part I: Overview, *ORSA Journal of Computing* 2(2)(1990), 180-206.

[24] Jones, C.V., Developments in Graph-based Modeling for Decision Support, *Decision Support Systems* 13(1995), 61-74.

[25] Kim, H.D., A Structured Markup Language for the Object-Oriented Repre-

sentation and Management of Decision Models on the Web, Forthcoming in *Decision Support Systems*.

[26] Kim, J.W., *A Modeling Framework for Integrating Decision Models and Information Systems*, Ph.D. Dissertation, KAIST, TaeJon, Republic of Korea, 1995.

[27] Kim, J.W., H.D. Kim, S.J. Park, Multi-facetted Approach to Mathematical Model Representation and Management, *Journal of the KORMS*, 23(2)(1998) (in Korean).

[28] Krishnan, R., PDM: A Knowledge-based Tool for Model Construction, *Decision Support Systems* 7(4)(1991), 301-314.

[29] Kwon, O.B. and S.J. Park, RMT: A Modeling Support System for Model Reuse, *Decision Support Systems* 16(2)(1996), 131-154.

[30] Lee, J.K. and M.Y. Kim, Knowledge-assisted Optimization Model Formulation: UNIK-OPT, *Decision Support Systems* 13(2)(1995), 111-132.

[31] Lee, J.S. *Structure Frame Based Model Management System*, Unpublished Doctoral Dissertation, University of Penn., 1989.

[32] Liang, T.P., Reasoning for Automated Model Integration, *Applied Artificial Intelligence* 4(1990), 337-358.

[33] Liang, T.P. and B.R. Konsynski, Modeling by Analogy - Use of Analogical Reasoning in Model Management Systems, *Decision Support Systems* 9(1)(1993), 113-125.

[34] Microsoft, OLE DB, http://www.microsoft.com/data/oledb, 1998.

[35] Muhanna, W.A., An Object-Oriented Framework for Model Management and DSS Development, *Decision Support Systems* 9(1)(1993), 217-229.

[36] Park, S.J., H.D. Kim and J.W. Kim, A Hypertext-based Integrated Modeling Environment (HIME), *Proceeding of KORMS '93 Spring Conference* (1993), 237-242 (in Korean).

[37] Park, S.J. and H.D. Kim, Constraint-Based Metaview Approach for Modeling Environment Generation, *Decision Support Systems* 9(4)(1993), 325-248.

[38] Park, S.J., J.W. Kim and H.D. Kim, IMF: A Modeling Framework for Integrating Decision Models and Information Systems, Unpublished Working Paper, KAIST, TaeJon, Republic of Korea, 1997.

[39] Park, S.J., J.W. Kim and H.W. Kang, Heuristic Knowledge Representation of Production Scheduling: An Integrated Modeling Approach, Forthcoming in *Expert Systems with Applications*.

[40] PowerSoft, *PowerBuilder User Manual*, 1993.

[41] Ramirez, R.G. and E. Lin, Subscript-free Indexing in a Mathematical Programming Language, *Proceedings of the 26th Annual Hawaii International Conference on System Sciences* (1993), 424-433.

[42] Stohr, E.A. and B.R. Konsynski, *Information Systems and Decision Processes*, IEEE Computer Society Press, 1992.

[43] Tung, L., R.G. Ramirez, R.D. St.Louis, Model Integration in an Object-Oriented Model management System, *Proceedings of the 24th Annual Hawaii International Conference on System Sciences* (1991), 284-290.

[44] Turban, E. *Decision Support and Expert Systems(2nd Ed.)*, Maxwell Macmillan International Editions, 1990.

Appendix A. Example Problem

A steel company produces three products, band, coil, and plate at three mills at Gary at Indiana, Cleveland at Ohio, and Pittsburgh at Pennsylvania. The steel products are shipped from the mills to customers at seven locations, Framingham at Massachusetts, Detroit at Michigan, Lansing at Michigan, Windsor at Ontario, St. Louis at Missouri, Fremont at California, and Lafayette at Indiana. Each mill has limitation on product capability for each product. Also, the company should meet demand requirements for each customer. There are restrictions on the total shipments of products from an origin to a destination because of limited shipping capacity.

The company wants to build a transportation model to determine product transportation quantities in order to minimize total transportation cost. The unit transportation cost is different according to origin, destination, and product. The total cost from an origin to a destination is calculated as the sum of transportation cost for the products. The transportation cost for a product from an origin to a destination is calculated by multiplying unit transportation cost and transportation quantities. The following tables list the elemental detail data.

(1) Supply

	GARY	CLEV	PITT
Bands	400	700	800
Coils	800	1600	1800
Plates	200	300	300

(2) Demand

	FRA	DET	LAN	WIN	STL	FRE	LAF
Bands	300	300	100	75	650	225	250
Coils	500	750	400	250	950	850	500
Plates	100	100	0	50	200	100	250

(3) Transportation Costs of Bands

	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	30	10	8	10	11	71	6
CLEV	22	7	10	7	21	82	13
PITT	19	11	12	10	25	83	15

(4) Transportation Costs of Coils

	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	39	14	11	14	16	82	8
CLEV	27	9	12	9	26	95	17
PITT	24	14	17	13	28	99	20

(5) Transportation Costs of Plates

	FRA	DET	LAN	WIN	STL	FRE	LAF
GARY	41	15	12	16	17	86	8
CLEV	29	9	13	9	28	99	18
PITT	26	14	17	13	31	104	20