경영정보학연구
제8권 제2호
1998년 9월

A Structured Markup Language for the Object-Oriented Representation and Management of Decision Models on the Web

Kim, Hyoungdo*

웹상에서의 의사결정모형의 객체지향적 표현과 관리를 위한 구조적 마크업 언어

김 형 도

The explosive growth of the Web is providing end-users access to ever-increasing volumes of information. The resources of legacy systems and relational databases have also been made available to the Web browser, which has become an essential business tool. Recently, model management on the Internet/Web is also proposed with its conceptual design or prototypical system like DecisionNet and DSS Web. However, they are also suffering from the same symptoms as the Web. Although we can identify the elements of a page with HTML tags and declare the relationships among the various document elements, they are semantically opaque to computer systems and have no domain-specific meaning. However, HTML is not extensible, so developers are forced to invent convoluted, non-standard solutions for embedding and parsing data. Extensible Markup Language (XML) is a simplified subset of SGML that has many benefits for folks who want to improve structure, maintainability, searchability, presentation, and other aspects of their document management. This paper proposes a structured markup language for model representation and management on the Web as an XML application. The language is based on a conceptual modeling framework, Object-Oriented Structured Modeling (OOSM), which is an extension of the structured modeling

---

* Internet Technology Team, Dacom R&D Center

I. Introduction

In the fields of MS/OR and Decision Support Systems (DSS), modeling processes are knowledge-intensive and time-consuming. Researches on Modeling Environments (ME) [Geoffrion, 1989b; Park and Kim, 1993] and Model/Modelbase Management Systems (MMS) [Bhargava et al., 1993; Blanning, 1985; Huh, 1993; Jones 1990; Kwon and Park, 1996; Muhanna, 1993] are active in order to support the modeling processes and related activities. To implement such an ME or MMS, a conceptual modeling framework is required for representing and managing decision models. Some distinctive frameworks for the purpose are as follows: structured modeling [Geoffrion, 1987], logic-based modeling [Bhargava et al., 1993; Krishnan, 1991], graph grammar [Jones, 1990], object-oriented modeling [Huh, 1993; Muhanna, 1993] and frame-based modeling [Binbasioglu and Jarke, 1986; Lee and Kim, 1995; Lee 1989]. They raises modeling to a higher plane of abstraction and generality compared with traditional solvers.

The explosive growth of the WWW is providing millions of end-users access to ever-increasing volumes of information. The resources of legacy systems and relational databases have also been made available to the Web browser, which has become an essential business tool. Recently, model management on the Internet/Web is also proposed with its conceptual design or prototypical system. DecisionNet [Bhargava et al., 1997] is a distributed, Web-based electronic market for decision technologies such as data, models, solvers and ME's. Consumer-provider interactions are facilitated by model management software agents. Problem-specific input and output data is exchanged via HTML forms, e-mail, or the Internet's file transfer protocols. In the DecisionNet, however, all execution occurs on the provider's machines, directed by agents on the basis of meta-information obtained during resource registration. Although it has the advantage of providing additional centralized services such as billing, consumers just view the results on the Web or gets a results file through the Internet. Furthermore, consumers have to be acquainted with the knowledge about modeling languages such as AMPL and GAMS.

DSS Web [MicroStrategy, 1998] enables users to access data warehouse and decision support applications from standard Internet browsers. It provides Internet users with the capability to perform advanced OLAP analyses directly against data warehouses. Custom applications can be implemented using Java/JavaScript compliant development tools. However, it is just a relational OLAP tool that analyzes and visualizes data on the Web. Representation and management problems about decision models are not considered.

Because a model is accessible by any model consumers or applications that have access rights on an open network such as the Internet, we need to develop a standard modeling language or an extensible framework to define the syntactic and semantic meanings of respective languages on the Web. The former is like the Structured Modeling Language [Geoffrion, 1992a; 1992b], and the latter the meta-view approach to the development of DSS modeling environment [Kim, 1992] in closed local environment.

The Web has also been suffering from the same symtoms. Although we can identify the elements of a page with HTML tags and declare the relationships among the various document elements, they are semantically opaque to computer systems. They have no domain-specific meaning. However, HTML is not extensible, so developers are forced to invent convoluted, non-standard solutions for embedding and parsing data, often contained within the HTML comment tag [Carlson, 1997]. Extensible Markup Language (XML) [Bray et al., 1998] is a simplified subset of SGML [ISO, 1986] that maintains the SGML features of validation, structure, and extensibility. XML is making rapid progress through standardization process. It has many benefits for folks who want to improve structure, maintainability, searchability, presentation, and other aspects of their document management. In addition to modifying the syntax and semantics of document tag annotations, XML also changes our linking model by allowing authors to specify different types of document relationships. Many communities have struggled to codify the tacit knowledge of their data. Even though the XML specification is not finalized, several other standards such as CML (Chemical Markup Language) [Murray-Rust, 1997], MathML (Mathematical Markup Language) [Ion and Miner, 1998], CDF (Channel Definition Format) [Ellerman, 1997], OSD (Open Software Description Format) [Hoff, 1997] are being proposed. Furthermore, a general approach to define meta-data about document structures is provided by both XML-Data [Layman et al., 1998] and Meta Content Framework (MCF) [Guha and Bray, 1997]. Several efforts are also underway to propose a base set of schemas for use with XML. RDF (Resource Description Format) [Lassila, 1998] Core Schema is a common set of properties that may be optionally used by all other meta-data schemas.

This paper proposes a structured markup language for model representation and management on the Web as an XML application. The language is based on a conceptual modeling framework, Object-Oriented Structured Modeling (OOSM), which is an extension of the structured modeling. In contrast to the previous object-oriented implementations of the structured modeling, it is an object-oriented extension to the structured modeling framework itself, where object-oriented concepts and structuring principles such as object-oriented modular structures, models as entities and specialization are supported.

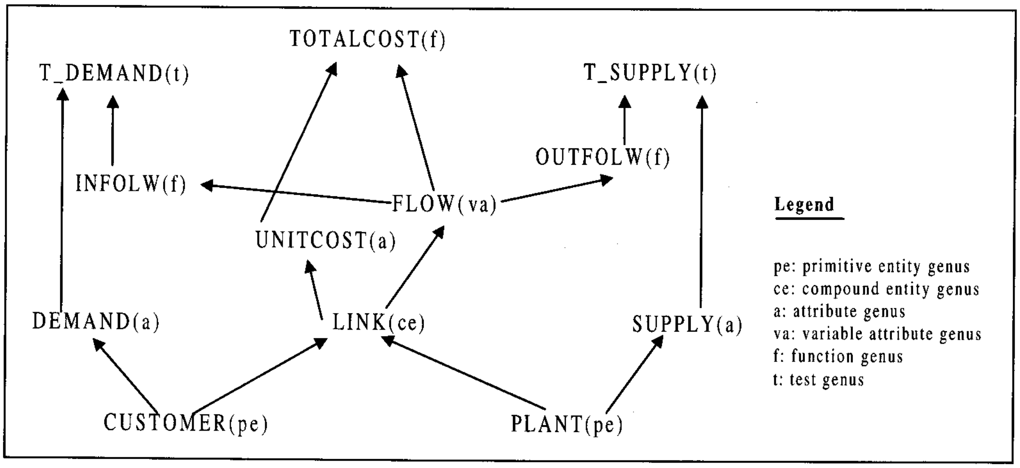
The rest of the paper is organized as follows. Section 2 introduces the OOSM framework with an example which is used illustratively throughout the paper. Following that, in section 3, an overview of the OOSM modeling language is provided. Section 4 discusses the advantage and disadvantage of the language. Finally, future research directions are summarized in section 5.

Ⅱ. Object-oriented Structured Modeling

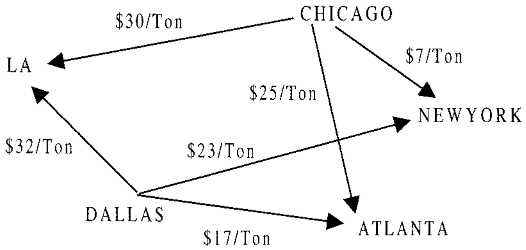
2.1 Structured Modeling

The structured modeling is a unified modeling framework based on acyclic, attributed graphs to represent cross-references between elements of a model and hierarchies to represent levels of abstraction. A model schema is primarily defined in terms of genera which group a set of data elements based on definitional similarity. There are five genus types: primitive entity, compound entity, attribute including variable attribute, function, and test. Primitive entities are existential in nature. Compound entities refer other entities already defined and have no value. Attributes associate a certain property and value with an entity or combination of entities. Variable attributes are attributes that must be determined by the model. Functions can have a value depending on those of other functions or attributes. Their values are respectively determined by a generic rule. Tests are like functions except that their values must be either true or false. Each genus except primitive entity genera has calling sequences, which identify its definitional dependencies on other genera. The definitional dependency between genera is monotonically ordered so that any cyclic dependency is not allowed. Genera are grouped into modules based on their conceptual similarity. The result of grouping is the modular structure all of whose leaves are genera, and all of whose non-terminal modes are modules. It allows users to view the model at different levels of abstraction.

Let's take the Hitchcock-Koopmans transportation problem [Dolk, 1988; Geoffrion, 1987; Lenard, 1987] as an illustrative example. The conceptual components can be depicted as a genus graph of <figure 1>. There are two primitive entity genera CUSTOMER and PLANT corresponding to consumer and producer of a product respectively. Each CUSTOMER has minimum DEMAND quantity and each PLANT has maximum SUPPLY quantity. There is one compound entity genus LINK corresponding to transportation route. Each LINK has UNITCOST for transporting one unit from a PLANT to a CUSTOMER and FLOW quantity to be determined by the model. A sample elemental details are depicted in <figure 2>.



<Figure 1> Genus Graph of the Transportation Problem



PLANT	SUPPLY
DALLAS	20,000
CHICAGO	42,000

CUSTOMER	DEMAND
NEWYORK	25,000
ATLANTA	15,000
LA	22,000

<Figure 2> Sample Elemental Detail of the Transportation Problem

2.2 Object-oriented Modular Structure

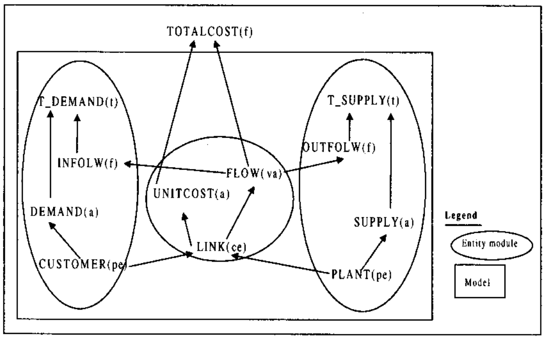
In object-oriented systems, attributes are tightly coupled with objects to order facts around objects and to guide conceptual views. Within the structured modeling, similar effects can be achieved by constraining the modular structure because it organizes generic structure hierarchically to manage the complexity of a model in terms of higher-order abstractions. Indexing properties also help to identify the ownership of each attribute-type genus: attribute, variable attribute, function, and test.

Object-oriented modular structure is then defined by the following criteria:

- Each (variable) attribute genus is restricted to call just one entity genus. Any (variable) attribute genus calling more than one entity genus should/could be modeled equivalently by another attribute genus calling one compound entity genus, which in turn calls all the entity genera called by the attribute genus. This forms one condition for structured modeling normal form [Farn 1985]. The (variable) attribute genus is then interpreted as an attribute of the entity genus.
- Any module is composed of only one entity genus and other attribute-type genera. Such modules are called entity modules.
- The entity modules must be contained in the model, that is, a model is only modularized around entity genera.
- The object-oriented modular structure embraces the role of indexing properties because the entity module relates each attribute-type genus with an entity genus.

<Figure 3> shows how to compose entity modules and models in OOSM of the example problem using a genus graph. Monotone ordering of a modular structure is listing genera in such an order that no element in a genus ever calls an element in a genus that is farther down the list. Note that you may lose the monotone ordering of a modular structure by imposing the object-oriented ordering principle. The worst case

may be acceptable if the conceptual validity and concise generation of object-oriented modular structure outweigh the benefits of monotonic modular structure. <Figure 4> defines the modular structure of the genus graph of <figure 3>.



<Figure 3> OOSM Genus Graph

```
&TRANSPORTATIONMODEL
  &PLANT
     PLANTi /pe/
     SUPPLY(PLANTi) /a/ {PLANT} :R+
     OUTFLOW( PLANTi) /f/ {PLANT} ;SUMj(FLOWij)
    T_SUPPLY(OUTFLOWi,SUPPLYi) /t/ {PLANT} ;
   OUTFLOWi<=SUPPLYi &CUSTOMER
     CUSTOMERj /pe/
     DEMAND(CUSTOMERj) /a/ {CUSTOMER} :R+
     INFLOW( PLANTj) /f/ {CUSTOMER} ;SUMi(FLOWij)
    T_DEMAND(INFLOWj,DEMANDj) /t/ {CUSTOMER} ;
   INFLOWj=DEMj&LINK
     LINK( PLANTi,CUSTOMERj) /ce/
     FLOW( LINKij) /va/ {LINK} :R+
     UNITCOST(LINKij) /a/ {LINK} :R
  TOTALCOST /f/ ;SUMiSUMj(UNITCOSTij * FLOWij)
```

<Figure 4> Object-Oriented Modular Structure

2.3 Models as Entities

Some function and test genera correspond to the attributes of a model. It is also desirable to specify various model characteristics for model management and decision support [Chari and Krishnan, 1990]. The name of modeler, the day of last update, etc. are added to such attribute list. There are some cases where various models are correlated and need to be integrated. Such relationships and attributes compose the knowledge for model management. In the domain of simulation, for example, system entity structure [Kim et al., 1990] provides the facilities for representing the knowledge.

To specify such knowledge about models, it is needed to view models as entities. In terms of structured modeling, any attribute-type genus can call the model, which is described by the genus. Such an attribute-type genus should be contained in the model. Note that any function or test genus calling the model must have another calling sequence within the model. Furthermore, a model and its characteristics can be contained in a module of another model. Namely, models can be structured hierarchically in mush the same principle of the structured modeling.

2.4 Specialization

Generalization/specialization concept of object-oriented systems is needed to efficiently represent and manage information via inheritance. In the structured modeling, a compound entity genus is a segmented tuple of primitive entity genera and/or other compound entity genera [Geoffrion, 1989a]. That is, a new entity genus dependent on other entity genera can be established. Such cases are generally defined in the form of relationship or association among them. Therefore, a compound entity genus is generally defined by two or more primitive entity and/or compound entity genera already defined. If a compound entity genus refers to only one, the former can be viewed as a specialization of the latter. For example, a product is either a final or in-process product, which has distinctive characteristics. It is natural and explicit to abstract common characteristics into product and to specialize respective characteristics into final or in-process product. The characteristics of product are then inheritable into the final and in-process products.

Ⅲ. OOSML: A Structured Markup Language for the OOSM

3.1 XML

Extensible Markup Language (XML) is a simplified subset of SGML that maintains the SGML features of validation, structure, and extensibility. SGML allows documents to be self-describing, through the specification of tag sets and the structural relationships between the tags. This specification is referred to as the Document Type Definition (DTD). HTML is a small hard-wired set of about 70 tags and 50 attributes, which allow HTML users to skip the self-describing aspect from a document. XML, on the other hand, retains the key SGML advantage of self-description through DTDs, while avoiding the complexity of full-blown SGML. XML is making rapid progress through standardization process. It has many benefits for folks who want to improve structure, maintainability, searchability, presentation, and other aspects of their document management. In addition to modifying the syntax and semantics of document tag annotations, XML also changes our linking model by allowing authors to specify different types of document relationships.

Many communities have struggled to codify the tacit knowledge of their data. Even though the XML specification is not finalized, several other standards are being proposed. CML (Chemical Markup Language) [Murray-Rust, 1997] uses XML to manage complex molecular information. MathML (Mathematical Markup Language) [Ion and Miner, 1998] is an XML application for describing the structure and content of mathematical expressions, allowing the markup of complex formulas, which mathematicians and computer scientists have been clamoring for since the earliest days of HTML. CDF (Channel Definition Format) [Ellerman, 1997] permits a Web publisher to offer frequently-updated collections of information from any Web server for automatic delivery. As an XML application, the CDF specification allows the publisher to specify the channels, the content available, the update schedule, and other information. For use in automated software distribution, OSD (Open Software Description Format) [Hoff, 1997] defines software packages and their interdependencies. A general approach to define meta-data about document structures is provided by both XML-Data [Layman et al., 1998] and Meta Content Framework (MCF) [Guha and Bray, 1997]. There are several efforts underway to propose a base set of schemas for use with XML. RDF (Resource Description Format) [Lassila, 1998] Core Schema is a common set of properties that may be optionally used by all other meta-data schemas. Web Interface Definition Language (WIDL) [Allen, 1997] is to describe the inputs and outputs of programs on the Web. It is a meta-data syntax implemented in XML that defines application programming interfaces to data and services, enabling automatic and structured Web access by compatible client programs, including mainstream business applications.

Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements by an empty-element tag. Each element has a type, identified by name (sometimes called its generic identifier or GI), and may have a set of attributes. Each attribute has a name and a value. Refer to Bray et al. [1989] for XML details.

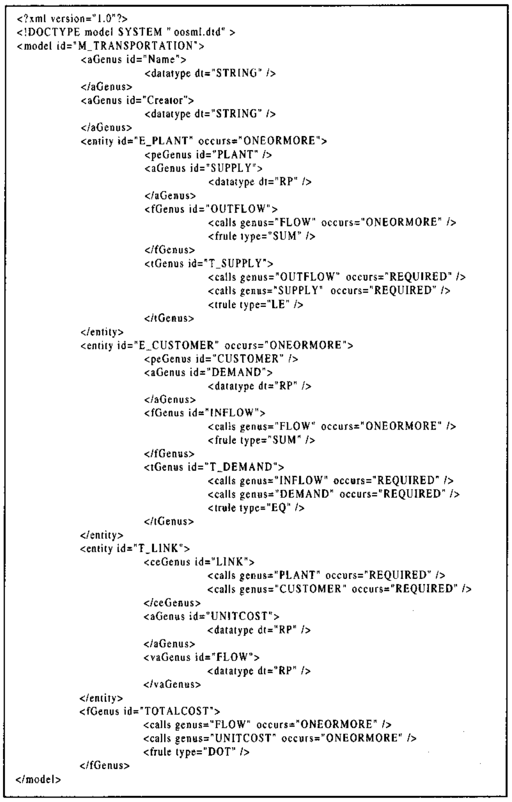
3.2 OOSML

For the specification of the OOSM models, this paper proposes a structured markup language, OOSML, based on the XML. An XML DTD for the OOSML is specified in <Appendix A>. The language will be illustrated using the transportation problem whose full specification is listed in <figure 5>. First of all, a model specification starts with a 'model' element, which is the target model to be described. For example, the transportation model is described by the following markrup.

```
<model id="M_TRANSPORTATION">
        ...
</model>
```

The "id" attribute serves a dual role of identifying the definition, and also naming the specific model class.

A model is an aggregate of other models and entities. The model "M_TRANSPORTATION" is an aggregate of "E_PLANT", "E_CUSTOMER" and "E_LINK". Furthermore, such a model can be described by attribute genus(aGenus), variable attribute genus(vaGenus), function genus(fGenus) and test genus(tGenus) like any entities. "M_TRANSPORTATION" has "Name" and "Creator" attribute genera.



```
<?xml version="1.0"?>
<!DOCTYPE model SYSTEM " oosml.dtd" >
<model id="M_TRANSPORTATION">
        <aGenus id="Name">
                <datatype dt="STRING" />
        </aGenus>
        <aGenus id="Creator">
                <datatype dt="STRING" />
        </aGenus>
        <entity id="E_PLANT" occurs="ONEORMORE">
                <peGenus id="PLANT" />
                <aGenus id="SUPPLY">
                        <datatype dt="RP" />
                </aGenus>
                <fGenus id="OUTFLOW">
                        <calls genus="FLOW" occurs="ONEORMORE" />
                        <frule type="SUM" />
                </fGenus>
                <tGenus id="T_SUPPLY">
                        <calls genus="OUTFLOW" occurs="REQUIRED" />
                        <calls genus="SUPPLY" occurs="REQUIRED" />
                        <trule type="LE" />
                </tGenus>
        </entity>
        <entity id="E_CUSTOMER" occurs="ONEORMORE">
                <peGenus id="CUSTOMER" />
                <aGenus id="DEMAND">
                        <datatype dt="RP" />
                </aGenus>
                <fGenus id="INFLOW">
                        <calls genus="FLOW" occurs="ONEORMORE" />
                        <frule type="SUM" />
                </fGenus>
                <tGenus id="T_DEMAND">
                        <calls genus="INFLOW" occurs="REQUIRED" />
                        <calls genus="DEMAND" occurs="REQUIRED" />
                        <trule type="EQ" />
                </tGenus>
        </entity>
        <entity id="T_LINK">
                <ceGenus id="LINK">
                        <calls genus="PLANT" occurs="REQUIRED" />
                        <calls genus="CUSTOMER" occurs="REQUIRED" />
                </ceGenus>
                <aGenus id="UNITCOST">
                        <datatype dt="RP" />
                </aGenus>
                <vaGenus id="FLOW">
                        <datatype dt="RP" />
                </vaGenus>
        </entity>
        <fGenus id="TOTALCOST">
                <calls genus="FLOW" occurs="ONEORMORE" />
                <calls genus="UNITCOST" occurs="ONEORMORE" />
                <frule type="DOT" />
        </fGenus>
</model>
```

<Figure 5> An OOSML Representation of the Transportation Problem

Entities group aGenus, vaGenus, fGenus and tGenus around peGenus or ceGenus, so they have to contain only one entity genus. The entity "E_PLANT" contains four genera, one of which is a peGenus "PLANT". An entity may be required or optional for a model which contains it, and may occur multiple times, as indicated by its "occurs" attribute having one of the four values "REQUIRED", "OPTIONAL", "ZEROORMORE" or "ONEORMORE". It has a default of "REQUIRED". One or more elements of the entity "E_PLANT" should occur in the instances of model "M_TRANSPORTATION". Attribute and variable attribute genera must have a data type of 'R', 'RP', 'I' or 'IP', where "R" stands for real values, "RP" for positive real values, "I" for integer values, "IP" for positive integer values. Attribute genus "SUPPLY" must have a positive real value.

Function and test genera can have "calls" elements which explicitly define mathematical dependency on other genera. The "genus" attribute of a "calls" element identifies the genus called by the element. The "calls" elements can also have an "occurs" attribute. Related with the mathematical dependencies, fGenus and tGenus must have a rule for computation or comparison, respectively. The rule of a function genus has a "type" attribute whose value is among "SCALE", "SHIFT", "POWER", "MIN", "MAX", "LN", "EXP", "VSUM", "VPROD", "SUM", "DOT" or "PROD". It has a default of "SUM". The function genus "OUTFLOW" has a rule type "SUM" that means summation on the "FLOW" genus. The rule of a test genus has a "type" attribute whose value is among "LE", "LT", "GE", "GT" or "EQ". It has a default of "LE". The function genus "T_SUPPLY" has a rule type "LE" that means "less than or equal to". In most systems and environments based on the structured modeling, mathematical knowledge is only embedded in strings, and may be interpreted or compiled for function evaluation. The specification of function genera by factorable functions [Lenard, 1986] is an early effort to conceptualize the generic rules in DSS area. Basic principle applied to the language design is to symbolically define the generic rules of function and test genera, and then to infer or constrain other facts based on the mathematical knowledge.

Ⅳ. General Discussion

This paper has adopted XML as a meta-language for specifying OOSML, of which characteristics are described using elements and attributes. Although HTML provides a universal way to present information, it only addresses the presentation of data. XML takes this one step further by addressing the context, or meaning of the data. By defining the structure of decision models with XML tags, finding, manipulating, acting on and interacting with the models are much easier. When a user's agent gets a model that conforms to the syntax and semantics of OOSML, it is able to analyze or manipulate the model easily and correctly. The highly structured delivery of data enables open interchange between servers and clients, and potentially between servers themselves.

Using a model schema represented by the OOSML, a user agent can perform diverse model management activities including form generation for modeling instances interactively on the Web, model instance generation from legacy software or databases, genus graph generation and so forth. The highly structured delivery of data enables the agent to present different views of the same information. For example, <figure 6> shows a model instance definition marked up by the identifiers in the OOSML definition (i.e., <figure 5>) of the transportation problem. The agent is able to support interactive modeling or automatic generation of such an instance.

```
<M_TRANSPORTATION>
        <Name>KoreaDistributionModel</Name>
        <Creator>H.D. Kim</Creator>
        <E_PLANT>
                <PLANT>DALLAS</PLANT>
                <SUPPLY>20,000</SUPPLY>
        </E_PLANT>
        <E_PLANT>
                <PLANT>CHICAGO</PLANT>
                <SUPPLY>42,000</SUPPLY>
        </E_PLANT>
        <E_CUSTOMER>
                <CUSTOMRER>NEWYORK</CUSTOMRER>
                <DEMAND>25,000</DEMAND>
        </E_CUSTOMER>
        <E_CUSTOMER>
                <CUSTOMRER>ATLANTA</CUSTOMRER>
                <DEMAND>15,000</DEMAND>
        </E_CUSTOMER>
        <E_CUSTOMER>
                <CUSTOMRER>LA</CUSTOMRER>
                <DEMAND>22,000</DEMAND>
        </E_CUSTOMER>
        <E_LINK>
                <LINK><PLANT>DALLAS </PLANT>
                        <CUSTOMER>NEWYORK</CUSTOMER>
                        <UNITCOST>23</UNITCOST>
                </LINK>
        </E_LINK>
        <E_LINK>
                <LINK><PLANT>DALLAS</PLANT>
                        <CUSTOMER>ATLANTA</CUSTOMER>
                        <UNITCOST>17</UNITCOST>
                </LINK>
        </E_LINK>
        <E_LINK>
                <LINK><PLANT>DALLAS</PLANT>
                        <CUSTOMER>LA</CUSTOMER>
                        <UNITCOST>32</UNITCOST>
                </LINK>
        </E_LINK>
        <E_LINK>
                <LINK><PLANT>CHICAGO</PLANT>
                        <CUSTOMER>NEWYORK</CUSTOMER>
                        <UNITCOST>7</UNITCOST>
                </LINK>
        </E_LINK>
        <E_LINK>
                <LINK><PLANT>CHICAGO</PLANT>
                        <CUSTOMER>ATLANTA</CUSTOMER>
                        <UNITCOST>23</UNITCOST>
                </LINK>
        </E_LINK>
        <E_LINK>
                <LINK><PLANT>CHICAGO</PLANT>
                        <CUSTOMER>LA</CUSTOMER>
                        <UNITCOST>30</UNITCOST>
                </LINK>
        </E_LINK>
</M_TRANSPORTATION>
```

<Figure 6> A Model Instance Definition of the Transportation Problem

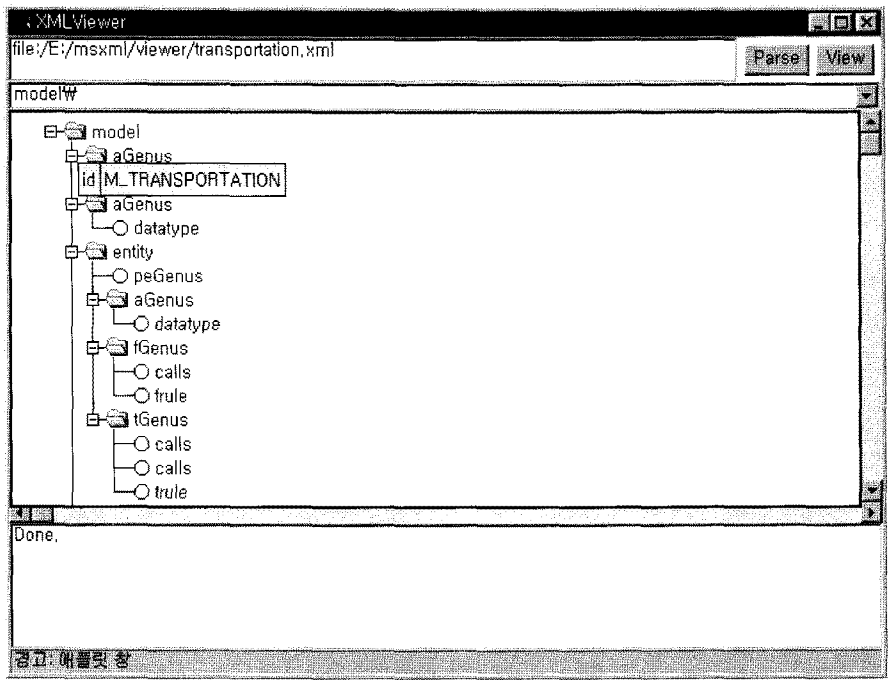
In the DecisionNet, a software agent leads a user through a session in which (s)he supplies requested data through a series of HTML forms. All the model manipulation and output generation are performed on the server side. This approach limits the role of clients or other servers. One reason for the limitation is related with the automatic generation of model instances from corporate systems and databases. Because a user agent can't interpret the meaning of the HTML forms, especially without model schema definition, the existing systems on the Web only support model-specific solutions or require that users should have high-level knowledge about modeling language specifics. In the DecisionNet, for example, users have to develop AMPL files. On the other hand, an XML application, WIDL (Web Interface Definition Language) [Allen, 1997], enables automation of all interactions with HTML/XML documents and forms, providing a general method of representing request/response interactions over standard web protocols, and allowing the Web to be utilized as a universal integration platform. It enables interfaces to be described for web sites that are not controlled by calling programs. Instead of interface description, OOSML is used for specifying decision models, whose structure and semantics are contained in its DTD.

One of the desirable features of a new generation of modeling systems called for by Geoffrion [1987] is the independence of model representation and model solution, with model interface standards to facilitate building a library of models and easily accessed solvers for retrieval, systems of simultaneous equations, optimization, and other important manipulations. Logically, two servers for modeling and solving problems may be separate, but the existing .systems on the Web is tightly coupled with specific solvers. The OOSML is a standard language for promoting the open trading of models between the two servers.

Note that OOSML is an XML application based on the object-oriented structured modeling framework. Although OOSML is semantically defined by a DTD, we need to standardize a language for decision model representation and management. The standard language will be a catalyst in trading or integrating models or model-related activities on the Web. The shift from structural HTML markup to semantic XML markup is a critical phase in the struggle to transform the Web from a global information space into a universal knowledge network [Khare and Rifkin, 1997].

V. Concluding Remarks

This paper proposes a structured markup language for model representation and management on the Web as an XML application. The language is based on a conceptual modeling framework, OOSM, which is an object-oriented extension of the structured modeling. The language supports object-oriented concepts such as object-oriented modular structure, models as entities and specialization using structured markups. Such a language will be a catalyst in trading or integrating models or model-related activities on the Web and transform the Web from a global information space into a universal knowledge network.



file:/E:/msxml/viewer/transportation.xml

model₩

```
model
  aGenus
    id  M_TRANSPORTATION
  aGenus
    datatype
  entity
    peGenus
    aGenus
      datatype
    fGenus
      calls
      frule
    tGenus
      calls
      calls
      trule
```

Done.

경고: 애플릿 창

<Figure 7> An XML Viewer Showing the Transportation Model in a Tree View

To date, the transportation problem defined by the OOSML have been checked by an applet viewer. <Figure 7> shows the viewer that uses a Java XML parser to display the XML document in a tree view. Shaded area in this figure lists all the attributes and values of an element pointed by the mouse cursor. There are many kinds of tools, based on such a parser, to support various activities such as specification, navigation and manipulation of XML documents. Model management tools for graphical model representation and integrating with solvers and information systems are expected to be developed.

Further research directions are as follows: a)to standardize such a modeling language through practical reviews and modifications in the DSS community, b)to implement access/analysis libraries to facilitate the integration or trading of decision models, and c)to implement automatic support functions for model formulation in the OOSM framework.

〈REFERENCES〉

[1] Allen, C., "WIDL: Application Integration with WIDL," http://www.webmethods.com/technology/widl.html, 1997.

[2] Bhagava, H.K. and S.O.Kimbrough, "On Embedded Languages for Model Management," *Decision Support Systems* Vol. 10 No. 3, 1993, pp. 277-299.

[3] Bhargava, H., R. Krishnan and S. Roehrig, "Model Management in Electronic Markets for Decision Technologies: A Software Agent Approach," *Proceedings of HICSS-30*, 1997.

[4] Binbasioglu, M. and M. Jarke, "Domain Specific DSS Tools for Knowledge-based Model Building," *Decision Support Systems* Vol. 2, No. 3, 1986, pp. 213-223.

[5] Blanning, R.W., "A Relational Framework for Join Implementation in Model Management Systems," *Decision Support Systems*, Vol. 1, No. 1, 1985, pp. 69-82.

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

[7] Carlson, D., "XML Documents Can Fit OO Apps," *Object Magazine*, Nov. 1997, pp. 24-26.

[8] Chari, S. and R. Krishnan, "Towards a Logical Reconstruction of Structured Modeling," *Proceedings of the 23rd Hawaii International Conference on System Sciences 3*, 1990, pp. 524-533.

[9] Dolk, D.R., "Model Management and Structured Modeling: The Role of an Information Resource Dictionary System," *Communications of the ACM*, Vol. 31 No. 6, 1988, pp. 704-718.

[10] Ellerman, C., "Channel Definition Format (CDF)," http://www.w3.org/TR/NOTE-CDFsubmit. html, March 1997.

[11] Farn, C.-K., *An Integrated Information System Architecture Based on Structured*

Modeling, Ph.D. Dissertation, University of California, Los Angeles, CA, 1985.

[12] Geoffrion, A.M., "An Introduction of Structured Modeling," *Management Science,* Vol. 33, No. 5, 1987, pp. 547-588.

[13] Geoffrion, A.M., "The Formal Aspects of Structured Modeling," *Operations Research,* Vol. 37, No. 1, 1989a, pp. 30-51.

[14] Geoffrion, A.M., "Computer-based Modeling Environments," *European Journal of Operational Research,* Vol. 41, 1989b, pp. 33-43.

[15] Geoffrion, A.M, "SML : A Model Definition Language for Structured Modeling: Level 1 and 2," *Operations Research* Vol. 40, No. 1, 1992a, pp. 38-57.

[16] Geoffrion, A.M., "SML : A Model Definition Language for Structured Modeling: Level 3 and 4," *Operations Research,* Vol. 40, No. 1, 1992b, pp. 58-75.

[17] Guha, R.V. and T. Bray, "Meta Content Framework Using XML," http://www.w3.org/TR/NOTE-MCF-XML, June 1997.

[18] Hoff, A., H. Partovi, T. Thai, "Open Software Description Format (OSD)," http://www.w3.org/TR/NOTE-OSD, August 1997.

[19] Huh, S.Y, "Modelbase Construction with Object-Oriented Constructs," *Decision Sciences,* Vol. 24, No. 2, 1993, pp. 409-434.

[20] Ion, P. and R. Miner, "Mathematical Markup Language," http://www.w3.org/TR/WD-math, January 1998.

[21] ISO, "Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML)," Standard 8879, 1986.

[22] Jones, C.V., "An Introduction to Graph-based Modeling Systems, Part I: Overview," *ORSA Journal of Computing* Vol. 2, No. 2, 1990, pp. 180-206.

[23] Khare, R. and A. Rifkin, "X Marks the Spot," http://www.textuality. com/sgml-erb/WD-xml-link.html, 1997.

[24] Kim, H.D., Metaview Approach to the Development of DSS Modeling Environments, Ph.D. Dissertation, KAIST, TaeJon, Republic of Korea, 1992.

[25] Kim, T.G., C.Lee, E.R. Christensen, and B.P. Zeigler, "System Entity Structuring and Model Base Management," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 20, No. 5, 1990, pp. 1013-1024.

[26] Krishnan, R., "PDM: A Knowledge-based Tool for Model Construction, "Decision *Support Systems* Vol. 7, No. 4, 1991, pp. 301-314.

[27] Kwon, O.B. and S.J. Park, "RMT: A Modeling Support System for Model Reuse," *Decision Support Systems,* Vol. 16, No. 2, 1996, pp. 131-154.

[28] Lassila, O., "Resource Description Framework (RDF)," http://www.w3.org/RDF, February 1998.

[29] Layman, A., E. Jung, E. Maler, H.S. Thompson, J. Paoli, J. Tigue, N.H. Mikula, S.D. Rose, "XML-Data," http://www.w3.org/TR/1998/NOTE-XML-data, January 1998.

[30] Lee, J.K. and M.Y. Kim, "Knowledge-assisted Optimization Model Formulation: UNIK-OPT," *Decision Support Systems,* Vol. 13, No. 2, 1995, pp. 111-132.

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

[32] Lenard, M.L., "Representing Models as

Data," *Journal of Management Information Systems,* Vol. 2, No. 4, 1986, pp. 36-48.

[33] Lenard, M.L., "An Object-oriented Approach to Model Management," *Proceedings of the 20th Annual Hawaii International Conference on System Sciences 3,* 1987, pp. 509-515.

[34] MicroStrategy, "MicroStrategy: A Leading Worldwide Provider of Enterprise DSS Software," http://www.strategy.com, 1998.

[35] Muhanna, W.A., "An Object-Oriented Framework for Model Management and DSS Development," *Decision Support Systems,* Vol. 9, No. 1, 1993, pp. 217-229.

[36] Murray-Rust, P., "Chemical Markup Language (CML) 1.0," http://www.venus.co.uk/omf/cml/, January 1997.

[37] Park, S. J. and H.D. Kim, "Constraint-Based Metaview Approach for Modeling Environment Generation," *Decision Support Systems,* Vol. 9, No. 4, 1993, pp. 325-248.

◆ 이 논문은 1998년 5월 12일 접수하여 1차 수정을 거쳐 1998년 8월 31일 게재확정되었습니다.

◆ 저자소개 ◆

김 형 도 (Kim, Hyoungdo)
현재 ㈜데이콤 종합연구소에서 멀티미디어 서비스와 EDI/EC분야에 관한 연구개발을 수행하고 있다. 서울대학교 산업공학과를 졸업(1985년) 하고, KAIST 경영과학과에서 석사(1987년) 및 박사학위(1992년) 를 취득하였다. 주요 관심분야는 의사결정지원, Systems Modeling & Simulation, 개인화된 멀티미디어 서비스, 전자상거래 등이다.

<Appendix A> An XML DTD for OOSML

```
<!ENTITY % nodeAttrs 'id ID #IMPLIED'>
<!ENTITY % elementAttrs 'occurs (REQUIRED|OPTIONAL|ONEORMORE|ZEROORMORE) "REQUIRED"'>

<!ELEMENT model (model|entity|aGenus|vaGenus|fGenus|tGenus)+>
<!ATTLIST model %nodeAttrs;>

<!ELEMENT entity ((peGenus|ceGenus),(aGenus|vaGenus|fGenus|tGenus)*)>
<!ATTLIST entity %nodeAttrs;
            %elementAttrs;>

<!ELEMENT peGenus EMPTY>
<!ATTLIST peGenus %nodeAttrs;>

<!ELEMENT ceGenus (calls)*>
<!ATTLIST ceGenus %nodeAttrs;>

<!ELEMENT calls EMPTY>
<!ATTLIST calls genus IDREF #REQUIRED
            %elementAttrs;>

<!ELEMENT aGenus (datatype)>
<!ATTLIST aGenus %nodeAttrs;>

<!ELEMENT vaGenus (datatype)>
<!ATTLIST vaGenus %nodeAttrs;>

<!ELEMENT datatype EMPTY>
<!ATTLIST datatype dt (R|RP|I|IP|STRING) "RP">

<!ELEMENT fGenus ((calls)+,frule)>
<!ATTLIST fGenus %nodeAttrs;>

<!ELEMENT frule EMPTY>
<!ATTLIST frule
            type (SCALE|SHIFT|POWER|MIN|MAX|LN|EXP|VSUM|VPROD|SUM|PROD|DOT) "SUM">

<!ELEMENT tGenus ((calls)+,trule)>
<!ATTLIST tGenus %nodeAttrs;>

<!ELEMENT trule EMPTY>
<!ATTLIST trule type (LE|LT|EQ|GT|GE) "LE">
```