Requirements of a Data Model for Modelbases

Thadthong Bhrammanee and Vilas Wuwongse

Computer Science & Information Management program, Asian Institute of Technology
{ttb, vw}@cs.ait.ac.th

Abstract

Although existing model representations attempt to give modelbases—a vital component of model-driven Decision Support Systems (DSS)—a powerful capability for storing decision models, most of them are prone to not supporting modeling life cycles and Web-based modeling incompatibility. More attention is thereby paid to improving the development of data models. Data model requirements are presented in an attempt to assist the development of a new modelbase representation and the improvement of existing ones.

1. Introduction

Modelbases—vital components of model-driven Decision Support Systems (DSS)—immensely benefit from efficient and expressive data models. The term “data model” is used here to refer to a particular language for modeling decision models. The term “decision model” represents a quantitative model used in Management Science and Operations Research (MS/OR). As decision models are a valuable organizational asset, their proper storage certainly promotes reuse and sharing, which thereby economizes efforts and the time spent on the modeling activities.

Recently, Internet and Web technologies have contributed to the design, implementation, and employment of modelbases in at least four ways. Firstly, there is a leap advancement of computation via the Web, i.e., server-side and client-side computation. This advancement, in particular server-side computation, makes model execution independent of a client’s platform. Secondly, an increasing proficiency of DSS technologies offers a greater variety of DSS tools (such as a mobile device) for storing or exploiting a model. Thirdly, a recent perspective of “Web as computer” publicizes the Web known as a “large repository for decision models”. Fourthly, model users are currently demanding a decision model for convenient use just as a service on the Internet [2]. Fortunately, the concept of Web services [22] is just in time to enable that realization of the demand of users. It offers services via the Web and is a new resource for creating value added business services.

Due to these four aspects, the modelbase community must reconsider the development of data models. There exist various efforts—either directly or indirectly related to design issues of data models—which offer certain important requirements which a data model should meet; unfortunately, none of those efforts point out them all, nor are sufficiently concerned with the problem of compatibility with Web environments. For example, Geoffrion [8] and Maturana [18] discuss design issues for mathematical programming, but specifically only for an algebraic modeling language. Bharadwaj et al. [1] point out phases in a modeling life cycle, but they emphasize a model management system. Hence, key requirements/considerations for a data model, applicable within the modern modeling environment, must be well analyzed and understood.

Section 2 reviews major approaches to decision model representation, Section 3 presents data model requirements and Section 4 draws conclusions.

2. Model representation overview

Current approaches to the development of data models can be grouped by their levels of interest in the structure of decision models and their specification techniques. Note that, generally speaking, a single representation framework may use a combination of various approaches and techniques.

Levels of interest in the structure of a decision model are driven by a concern regarding the context of a decision model, ranging from ignoring the algorithmic aspect to high emphasis on the executable system of a decision model. The first—data-centric—level employs
a traditional database data model (such as relational and hierarchical structures) as a design principle and does not contain an explicit representation of a computation relationship. It has the advantage that users are familiar with the data model under study [3,5]. The second—structure-centric—level presents more information of a decision model, so that users are able to understand the large picture of a model structure in terms of a definitional system as well as are able to specify more details of the problem context. It has the ability to group the sub-model or to make a module which creates a concise representation [7]. The third—abstraction-centric—level reduces the complexity and increases the efficiency of a model by hiding all but the relevant data. Some frameworks (e.g., [14] and [15]) fully adopt and utilize the mechanism of an object-oriented (OO) data model so that a wide range of operations can be performed on the model. The fourth—logic modeling-centric—level applies a logic-based theory as an underlying principle. Integrity constraints are generally employed to enforce a syntactic structure. An inference mechanism is employed for retrieving implicit derived information from the model specification [13]. However, it requires a user to learn a number of model vocabularies and grammars. The last—computation-centric—level represents a decision model close to the algebraic or algorithmic form, so that users can investigate and formulate a mathematical equation as well as easily execute the model. However, most frameworks in this group lack a formal data model; their model specifications are in plain English [6,7,10].

Another perspective to classify the design of a data model is by the form (or specification technique) specifying it. The first—graphic—form represents the relations between parts of the model graphically. Diagrams and icons are popular tools for creating a visual perception of a decision model [3,4,7,15,19]. The second—text—form is more descriptive than the graphical form since a use of graphics may face either space limitation or unavailability of a proper modeling environment. Some frameworks in this group use human-understanding text to explain certain details of a model construct, e.g., a plain text description in the “interpretation” part of [7]. In contrast, machine-understandable languages are widely used. A decision model may either utilize an existing programming language such as Prolog or issue its own syntax [6,7,10]. The third—algebraic—form represents a decision model in a way which is close to algebraic notation. It requires the use of symbolic subscripts. Algebraic representation is commonly used as part of text-based modeling. The last—schematic—form restricts the structure and content of a decision model to a certain schema which expresses shared vocabularies and rules. This approach emerges due to the increasing demand for exchanging a decision model in agreement with a common vocabulary. The schema also allows machine validation of document structure [19,11].

3. Data model requirements

A data model typically describes a conceptual schema, constraints and operations. Yet, the modern modeling environment demands convergence between model representation and Web practice, whence a data model also has to integrate the following salient characteristics:

Model creation/formulation support contributes to conversion of a problem description into a particular form in which either a human or a machine can perform an analytical task. Two popular methods for the formulation of a new decision model are, firstly, creation from scratch [4,6] and, secondly, creation from template [14,15,17]. Regardless of the model formulation method, a data model should be sufficiently scalable so that large and small problems as well as various problem types can be formulated. For this purpose, a data model should have a concise and sufficient notation which covers problem types and model elements, be able to hide detail of abstraction and possess a grouping mechanism.

In addition, a data model should provide a supporting tool suitable for different types of model users and assisting decision model conceptualization. However, those tools migrate toward a Web application, the bandwidth speed is a vital consideration in delivering the graphical content.

Model advertisement/registration support aims to create user awareness of an available decision model by announcing and providing public information of the model. In particular, the “Web as computer” era requires model features to be easily understood by remote users/applications (with a heterogeneous platform). Private and public conventions are generally two modes of model advertisement/registration.

For the private convention mode, decision models are advertised on the model owner’s model repository. Each repository has a distinct way to index/arrange models. In contrast, the public convention mode demands a decision model to register with respect to the universally agreed convention of external register services [21].

In addition, metamodel has a crucial role in model advertisement, whence the framework should have precise definitions of the constructs and rules needed for expressing a decision model. In particular, a metamodel
should explain well detailed information such as required inputs/outputs and the model access method.

**Model discovery/selection support** is the task of finding and eliciting a suitable decision model from a model repository. The data model should be designed to make it easy to apply a matching operator at various element levels.

At this stage, there are at least three commonly available model selection strategies. The simplest one makes a request from the tree structure of a problem type [9]. The next one selects a model at the input/output level. A relational schema may be employed by introducing into each model “input” and “output” relations. Finally, employment of a frame, another approach to deliver efficient model selection, enables exploration of internal detail of the decision model [17].

Nevertheless, should modelbases be integrated with the Web services environment, this requires that the data model of a decision model description adheres to or is compatible with a common accepted description language such as Web Services Description Language (WSDL) [23]. Such language describes the interface to Web applications in order to enable a proper query result.

**Model modification/customization support** enables a decision model to either be adjusted to fit an individual’s specification or be modified from other existing models.

Firstly, a data model is required to provide *ease of modification*, i.e., a data model should be so simple that users take little time to understand, learn as well as extend the model with minimum effort. However, a tradeoff of the lack of informative internal configuration of the decision model should be considered. Secondly, *integrity constraints* are rules which constrain valid states. A data model pertains to at least two kinds of constraint. The first one is that of the built-into data model itself. For example, specification of the connection of the inheritance class assures that any change in the parent is automatically passed on to all its children. The second constraint is application-dependent and allows storage of user-defined assertion.

Finally, a data model should enable *model transformation across the modeling environment*. The reasoning behind this is simple. Two different organizations sharing the same decision model might be able to afford different kinds of solvers. The employment of XML-base language to represent a decision model would promote an opportunity to transform a model across the modeling environment, since XML is currently a standard way of encoding both text and data across diverse platforms [11].

**Model composition/integration support**: Even though the terms model composition and model integration are often used interchangeably, Krishnan and Chari [12] point out the differences between these two terms, i.e., that the former is to link independent models without modifying any of them [3], while models in the latter are modified.

It is also essential to emphasize that data models should allow for embedding a conflict resolving mechanism. Three major model integration conflicts [12] are *naming conflict* (use of the same name to refer to individuals), *granularity conflict* (different granularity basis of models, e.g., compute annually or quarterly) and *dimensional/unit of measurement conflict*. Nevertheless, the literature shows that model integration involves extensively human intervention or determination (such as [7]) and it is a challenging research area of the model management community.

**Model execution support** relates to the process of generating results. A data model should support transformation from the model to the solution space, i.e., it should not burden the creation of the representation/notation which a model solver can understand. Frameworks in a computation-centric group, i.e., algebraic modeling languages, are executable and are understood by the solvers. Unfortunately, they have the poor meta-level of a data model. In contrast, frameworks which employ a high conceptual level are hardly executable. Nevertheless, Web technologies have affected the execution of decision models, whence frameworks should be ready to survive in a Web computing environment.

Web technologies provide flexible computation and have seen advances in both server-side and client-side computing [2]. Exploiting server-side computing is so fertile that it gives an organization, which does not own a model solver, an opportunity to execute a model. Even so, a solver algorithm residing in a remote machine requires a specific data structure, while a model can be represented in many forms such as graphical and textual models. Hence, the issue of executing a model without heavy transformation should be considered.

**Support of interoperability** is essential in a Web-enable modeling environment so that a data model must support communication among software and hardware on heterogeneous vendors and heterogeneous platforms. In particular, an activity on the Web services platform demands for sending messages plain XML to ensure interoperability, since XML is the current standard for data representation and interchange among various Web applications. Altogether, if a modeling activity should
migrate to a Web services framework, model representation should abide by the exchange standard.

**Support of representation of mathematical equations** is the ability of a data model to express mathematical notation, including algebraic formulae and symbols. The intent of presentation of a mathematical equation is to allow quantitative inspection and understanding of a decision problem in a quantitative manner. Although the high conceptual level of model representation provides the model user with an easily-understood practice (such as a graphical language), some types of model users—especially MS/OR model expert—find it to be insufficient. Those high conceptual ones are just a front-end to the underlying mathematical expression.

As the modeling environment moves toward the Web, an approach to data modeling may consider employment of XML-based markup language such as MathML to represent mathematical expressions on the Web. It is expected that there will be on the Web more engines which can understand mathematical expressions, especially in MathML format.

**Support of indexing** is “one of the critical operations a modeling support system must perform”, said Lazimy [15]. In any circumstance, a data model is required to support an indexing operation—the ability to characterize a set of data and allow performance of basic operations, i.e., declaration of index set, define index set and application of functions to an index set.

Use of index and subscript are mandatory in most algebraic modeling languages. They are exhaustively discussed in [18] and [8] as an important design issue of a modeling language for mathematical programming. Nevertheless, there are also frameworks which question the use of symbolic subscripts for indexing. Hence, Lin et al. [16] and Lazimy [15] propose a data model which can eliminate the use of subscripts by employment of relational database theory and object-oriented theory, respectively. In essence, these subscript-free modeling languages benefit the user in at least two ways, i.e., ease of model formulation and ease for whose not originally create the model to understand it.

Function on index set is another crucial operation. Common index set functions are “subset of”, “union”, “intersection” and “complement”. Yet, not all existing frameworks support well index representation requirement, i.e., even if they can represent the index set, it is not simple to apply functions to it [3,5,11].

**Support of representation of incomplete information**—one of the major challenges of model representation—is the ability to represent unknown information. Note that in this context, unknown information differs from a “variable” (such as a decision variable) which is a basic model element. It is the ability to form and allow storing a model even when some parts of the model (such as the objective function) are still needed.

This is essential, because there is a chance that modelbases can encounter incomplete information, for example, the type of coefficient is unknown. A lack of such information should not prevent users from composing a model. Representation of incomplete information has not been adequately addressed in the literature. However, the authors wish to argue that data model developers should consider this issue.

### 4. Conclusions

**Table 1. Excerpt of a comparison of representation frameworks to characteristics of data model**

<table>
<thead>
<tr>
<th>Approaches/frameworks</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Graphical view</td>
</tr>
<tr>
<td>Data-centric</td>
<td>Fourer [5]</td>
</tr>
<tr>
<td>Structure-centric</td>
<td>OOSML [11]</td>
</tr>
<tr>
<td>Abstraction-centric</td>
<td>RMT [14]</td>
</tr>
<tr>
<td>Logic-centric</td>
<td>PM* [13]</td>
</tr>
<tr>
<td>Computation-centric</td>
<td>LPL [10]</td>
</tr>
</tbody>
</table>

*Employ a Table metaphor **Require extensive transformation to do so
Although a diversity of data modeling frameworks exist in the modelbase community, a more expressive conceptual data model is still lacking. The need for it is driven by the impact of Internet and Web technologies. Data model requirements—a check list to determine whether a framework apprehends essential characteristics of a good data model—have been presented.

As a result of comparisons, no single framework wins general overall acceptance. (Table I provides an except of the comparison). In order to satisfy the requirements, either a new data model must be developed or an existing framework could produce a specialized representation of itself, in order to be compatible with the modern modeling environment.

In conclusion, since the complexity of the Web, which influences modelbases, will continue to increase, exploitation of an expressive data model is a necessity. The proposed data model requirements are therefore a first step toward the development of a new breed of representation framework for modelbases.

5. References


[21] Universal Description, Discovery and Integration (UDDI), http://www.uddi.org


[23] Web Services Description Language (WSDL), http://w3.org/TR/wsd1