Towards a Representation Framework for Modelbases

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

Abstract
This XML-based representation framework for modelbases supports a modeling life cycle as well as promotes shareable and reusable decision models over the Web. It comprises two representation layers: Model Ontology and Model Schema, the first of which plays a crucial role in defining an agreement for the use of concepts among representation layers as well as users/applications of modelbases. Model Schema provides a general description for each set of common decisions. Supportive capabilities of the proposed representation framework demonstrate its adequacy.

1. Introduction
The newly developed representation framework for modelbases is strongly motivated by the need for convenient use of a decision model across the Web as well as promotion of model reuse. A modelbase—which stores decision models—is a vital component of model-driven Decision Support Systems (DSS). The term “decision model” refers to a quantitative model used in Management Science and Operations Research (MS/OR).

There exists much research on the representation framework for modelbases. From the perspective of a decision model structure, there are five approaches to model representation [4]—ranging from those which ignore the algorithmic aspect to those with high emphasis on executable systems. The data-centric approach employs a traditional database data model as its design principle, the structure-centric approach views a decision model in terms of a definitional system, the abstraction-centric approach aims to reduce the complexity of models by hiding all but relevant data, the logic-centric approach is based on logic-based theory and the computation-centric approach presents an algebraic form of a decision model and aims to provide an executable modeling language.

On the other hand, from the research perspective on the forms specifying decision models, there are at least four approaches to represent them [4]: The graphic approach graphically represents the relations among parts of a decision model, the text approach provides a textual representation of a decision model, the algebraic approach represents a decision model in a way which is close to an algebraic representation and the schematic approach restricts the structure and content of a decision model to a certain schema. Note that various approaches can be combined in a single representation framework.

Up to the present time, there exist at least three major aspects which affect the development of a modelbase framework:

User aspect- a user’s unawareness of an available decision model on the Internet and incompatibility of the user’s machine [3].

Technologies- a frog leap advancement of computation via the Web, an increasing proficiency of DSS technologies such as mobile devices and the emerging Web services framework [18].

New business model- value added business services which make a decision model available for seven days and twenty-four hours.

Those aspects clearly display that the Internet is changing the way of exploiting decision models. Unfortunately, current representation frameworks are neither adequately Web-accessible nor sufficiently flexible to exchange models across platforms [4].

The proposed new framework employs XML-based representation and thus facilitates model exchange throughout the web as well as promotes interoperability, simplicity and extensibility. It comprises two representation layers: Model Ontology and Model Schema. The first provides a shared terminology which users or applications of modelbases can jointly understand and employ; the second contains a model description of a generic decision model. A model instance can be later represented as a specialization of a model described in the Model Schema layer. All layers are seamlessly interoperated by means of the Web
Ontology Language (OWL) [9] and RDF Declarative Description (RDD) (the Appendix provides a brief introduction to RDD) [1]. OWL provides a simple ontological-modeling facility. RDD enhances OWL expressive power with a computation mechanism and extends OWL elements with variables.

Sections 2 and 3 present Model Ontology and Model Schema, respectively. Section 4 discusses the framework’s capability and Section 5 concludes.

2. Model Ontology

Model Ontology is a central vehicle for the reuse of knowledge in modelbases. Purposes of Model Ontology are:

- Definition of the meaning of terms/concepts and provision of shared terminologies, e.g., the term “activity” refers to the concept which requires a “resource”
- Embedment of quantitative problem knowledge, e.g., the taxonomy of mathematical problems.
- Implementation of a set of axioms, e.g., a stochastic model always contains random variables. The axioms will enable the model ontology to find automatically answers to various questions regarding the decision model domain.

Existing ontologies related to MS/OR have some of the necessary concepts and relations which can serve as shared conceptualization of decision models. Unfortunately, none of them have all necessary ones. For example, GAMS [8] covers most of the top concepts of mathematical problem; however, it does not include many basic concepts of the real world domain such as “resource” and “activity”. Therefore, the newly developed Model Ontology reuses parts of existing ontologies and classification schemes. The related ontologies are:

GAMS [8]: The Guide to Available Mathematical Software (GAMS) is the NIST de facto tree-structured classification system for mathematical and statistical problems.

OZONE [15] provides a framework for constructing an appropriate domain model. Its application areas cover scheduling, manufacturing, space and transportation logistics. The five basic concepts in OZONE ontology are demand, product, resource, activity and constraint.

Process ontology [14]: The NIST standard of exchanging process information among manufacturing applications. The wide range of application areas which can employ this ontology include process planning, scheduling, business process reengineering, etc. Key concepts are activity, activity occurrence, time point, and object.

2.1 Concepts and properties in Model Ontology

Model Ontology comprises four main parts: Problem Taxonomy contains the taxonomy of decision problems. “Model” is the top level concept/class. Problems at the lower level of the hierarchy become subclasses. For example, “Transportation Problem” is a subclass of “Constrained Optimization” which is a subclass of “Optimization Problem”.

Mathematical Model Elements provide the indispensable elements of a quantitative model. Major elements are “Object”, with the subclasses “Variable” and “Index”, and “Relation” [7]. “Variable” has “Dependent Variable” and “Independent Variable” as subclasses. “Primitive Index” and “Compound Index” are kinds of “Index”. “Relation” has “Objective” and “Constraint” as subclasses. “Constraint” has “Arithmetic Constraint” and “Logical Constraint” as subclasses [7].

World Elements are objects in the real world which relate to a decision problem. They also correspond to Model Elements in some sense. “Time”, “Resource”, “Activity”, “Quantity”, “State”, and “Event” are examples of the subclasses of “World Element”. “Time” can be specialized to “Time Point” and “Time Interval”, “Resource” to “Consumable Resource” and “Reusable Resource”.

Relations, Functions, and Operators- A “Relation” is a property which denotes a link between model elements or entities, for example, “Equal” and “Less Than”. A “Function Type” denotes a predefined procedure, for example, “Sum” and “Mean”. An “Operator” denotes a symbol which operates between functions, for example, “Minus” and “Time”.

Enterprise ontology [17] is supported by various parties including IBM, and the UK Department of Trade and Industry. It aims to provide various parties—from business managers to software engineers—with a shared understanding of the aspects of a business enterprise. The major concepts in Enterprise ontology are activity, organization, strategy, marketing and time.
Figure 1 shows some of the concepts defined in Model Ontology. The Model Schema layer can use these concepts in a model description. The example illustrates the concepts related to a logistic domain. Here, “Resource” is an entity which supports or enables execution of “activity” [15]. “Shipment” is considered to be “Activity” in a logistic domain. An “Activity” uses a “Resource” to achieve or fulfill some “Events”. “Plant” is a kind of “Reusable Resource”. Each plant has a certain capacity, denoted as “Product Supply”, which points to “Supply Quantity”. “Plant” corresponds to a “Primitive Index”. Plant is also known as a shipping origin. “Expense” or money is a “Consumable Resource” consumed by a “Shipment”. “Shipping Cost” is a numerical quantity of the “Expense” of a “Shipment”.

In addition, the Model Ontology layer also stores Ontology axioms. They are specified by RDD clauses. Axioms provide rules, including structural knowledge of a decision model. A “structural knowledge” is referred to in Krishnan et al. [12] as “model construct knowledge” — an important component of knowledge-based model formulation.

### 3. Model Schema

Model Schema describes a generic decision model, whence the Model Schema layer stores various model descriptions. Every decision model has both internal and external views, defined in model description (Figure 2a): Model Profile serves as a black box view of a model, while Model Configuration serves as a glass box view describing the internal configuration of a model.

Different decision model classes (e.g., optimization and simulation) can define their skeleton—major model constructs—inside a “Construct” element. For example, model constructs of all models in the optimization problem class consist of objective function, constraint, decision variable and coefficient.

Consider the example of a transportation problem; it inherits all constructs of an optimization problem. However, a transportation problem extends the model description to be more specific to the problem by selectively drawing logistic-related concepts from the Model Ontology. Figure 2b depicts a fragment of the model “Construct” element of a generic transportation model. Clearly, detail semantic of each concept specified here is traceable in the Model Ontology. For example, “Plant” is a “Primitive Index” and “Resource”, “Route” is “Compound Index”, etc.

In addition, the definition of model can be specifically defined for a more specific problem situation by specialization of a generic model stored in the Model Schema layer. This allows different problems with a similar problem domain situation to share a single model schema.

### 4. The supportive capabilities

The framework is found to support essential data model characteristics for modelbases indicated in [4]. The term “data model” refers to a particular language for modeling a decision model. In summary:

**Model creation/formulation support:** Various users of modelbases [2] have different objectives and need different supports to formulate a model. The representation layers used here appropriately satisfy the needs of each user type. Model builders maintain the ontology so that model descriptions can be properly created by analysts. Decision makers can create a decision model with instances by supplying data to the model description. In addition, the framework provides a graphical view of a decision model by employment of an RDF graph [10], with some extensions to suit an OWL expression.
Model advertisement/registration support: A standard online register for Web services [16] is friendly with Web Services Description Language (WSDL) [19] and DAML Services (DAML-S) [5]. The proposed framework then provides some equivalent features to WSDL and DAML-S, in order to satisfy the standard model registry.

Model discovery/selection: The proposed framework employs an RDD clause in order to generate the selection result for a complex model selection. By encoding the model in RDD language (see example in Figure 3), the bodies of the clause specify the condition. The head drives a selected decision model.

Model modification/customization support: Firstly, support of problem type modification (Decision problems in the same hierarchy can be modified from one to another), secondly, support of model transformation across the modeling paradigms by employment of a transformation language such as Extensible Stylesheet Language Transformation (XSLT), finally, integrity constraint support by formalizing an integrity constraints as RDD clauses.

Model composition and integration support: Conflict resolution [11] is a requirement during performance of model composition/integration. By means of explicit specification of concepts which are stored in Model Ontology, naming conflicts can be solved. By means of integrity constraint support and the computation mechanism of RDD, granularity and dimensional/unit of measurement conflicts can be solved.

Model execution support and support of representation of mathematical equations: The framework actually allows representing MathML—a mutual way to provide a mathematical equation readable by human and machine—in the Model Schema. MathML representation of a decision model can be sent to a MathML-recognized solver for execution. Alternatively, XSLT can be applied to model description in order to transform an existing model into a representation of a modeling framework which owns a model solver such as AMPL [6].

Support of interoperability: Use of XML-based language to represent a decision model ensures interoperability, since XML is a de facto exchange standard among industries.

Support of indexing: The proposed framework is considered a symbolic subscript-free language, hence, provides ease of model formulation and ease for those who not originally create a model to understand it [13]. In addition, common index set functions define a dynamic subset (e.g., a subset of, union of, intersection of and complement of) are allowed.

Support of representation of incomplete information: By employment of RDD variables, the model containing unknown information (e.g., an unknown objective function and an unknown relation) can be formed as a non-ground RDD expression and stored in a model repository. Users can later on apply specialization to a model—when the information is known—in order to turn it into a complete model (ground RDD expression). Figure 2b shows an employment of RDD string-variable ($) to define the unknown “relation” as $S: Relation.

5. Conclusions

A new framework of model representation is proposed as a means of supporting modeling life cycle and handling current and future environmental impacts. The framework utilizes the ontological discipline to define terms which can be communicated across to modelbase users, including people and applications.

The merit of the metadata description facility—OWL—and the expressiveness of RDD drives a uniform representation of all three representation layers for modelbases. Thus, each layer is ready to exchange information items. In addition, the framework is feasible to integrate into a Web services framework and enjoy computation via the Web which also abates a user’s unawareness of decision models and machine configuration incompatibility problems. The development of a prototype XML-based model representation under the proposed framework is underway.

6. References


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


Appendix

RDF Declarative Description (RDD) [1] is an RDF-based knowledge representation, which extends ordinary well-formed RDF elements by incorporation of variables for an enhancement of expressive power and representation of implicit information into so called RDF expressions. Ordinary RDF elements – RDF expressions without variable – are called ground RDF expressions. Every component of an RDF expression can contain variables, e.g., its expression or a sequence of sub-expressions (E-variables), tag names or attribute names (N-variables), strings or literal contents (S-variables), pairs of attributes and values (P-variables) and some partial structures (I-variables). Every variable is prefixed by $T$: where $T$ denotes its type; for example, $S:value$ and $E:expression$ are S- and E-variables, which can be specialized into a string or a sequence of RDF expressions, respectively.

An RDD description is a set of RDF clauses of the form:

$$H \leftarrow B_1, \ldots, B_m, \beta_1, \ldots, \beta_n$$

where $m, n \geq 0$, $H$ and the $B_i$ are RDF expressions, and each of the $\beta_i$ is a predefined RDF constraint – useful for defining a restriction on RDF expressions or their components. The RDF expression $H$ is called the head, the set $\{B_1, \ldots, B_m, \beta_1, \ldots, \beta_n\}$ the body of the clause. When the body is empty, such a clause is referred to as an RDF unit clause and the symbol ‘$\leftarrow$’ will often be omitted; hence, an RDF element or document can be mapped directly onto a ground RDF unit clause. Given an RDD description $D$, its meaning is the set of all RDF elements which are directly described by and are derivable from the unit and non-unit clauses in $D$, respectively.