18th European Symposium on Computer Aided Process Engineering – ESCAPE 18
Bertrand Braunschweig and Xavier Joulia (Editors)

OntoMODEL: Ontological Mathematical Modeling Knowledge Management

Pradeep Suresh[a], Girish Joglekar[a], Shuohuan Hsu[a], Pavan Akkisetty[a], Leaelaf Hailemariam[a], Ankur Jain[b], Gintaras Reklaitis[a] , and Venkat Venkatasubramanian[a1]

[a]*School of Chemical Engineering, Purdue University, West Lafayette IN 47907, USA*
[b]*Enterprise Optimization, United Airlines, Chicago IL 60007, USA*

Abstract

In this paper we describe OntoMODEL, an ontological mathematical model management tool that facilitates systematic, standardizable methods for model storage, use and solving. While the declarative knowledge in mathematical models has been captured using ontologies, the procedural knowledge required for solving these models has been handled by commercially available scientific computing software such as Mathematica and an execution engine written in Java. The interactions involved are well established and the approach is intuitive, therefore not requiring model user familiarity with any particular programming language or modeling software. Apart from this key benefit, the fact that OntoMODEL lends itself to more advanced applications such as model based fault diagnosis, model predictive control, process optimization, knowledge based decision making and process flowsheet simulation makes it an indispensable tool in the intelligent automation of process operations. This paper also discusses the shortcomings of existing approaches that OntoMODEL addresses and also details its framework and use.

Keywords: Ontology, Mathematical modeling knowledge, knowledge management.

1. Introduction

Mathematical knowledge is a very broad term that could be used to describe various components of mathematics such as theorems, lemmas, proofs etc. In this work we use the term 'Mathematical Modeling Knowledge' to denote the vast amount of knowledge that exists as mathematical models, the modeling assumptions and the model solving procedures associated with them. Compared to other forms of knowledge, like rules, decision trees, guidelines etc., mathematical knowledge is more abstract and highly structured (Farmer 2004). Most forms of mathematical knowledge are either embedded in specific software tools such as unit operation models in simulation software, or have to be entered into a more general mathematical tool following a specific syntax, like Matlab or Mathematica. Much of this knowledge, however, concerns specific applications and expressed procedurally rather than declaratively. For example, in the application domain of chemical process development, Aspen Custom Modeler (See URL) uses a specific modeling language for user to provide or create new models in order to be used with other Aspen products. The need for an automated, systematic, reusable mathematical model knowledge capturing environment is very real and justified in the context of the large amounts of process and product information and

[1] Corresponding author

knowledge generated and stored. In this work we will propose an ontological approach to address these needs.

2. Existing Modeling Approaches and Limitations

In almost all of the mathematical modeling knowledge management approaches mentioned below, the model is unique to the format it is created in and hence cannot be shared/reused, and even if so, interfacing is neither user friendly nor transparent.

2.1. Microsoft Excel

Perhaps the most popular approach to handle simple mathematical models is Microsoft Office's spreadsheet application, Excel. It provides basic calculation and plotting abilities and can copy formulae and refer values based on cell names. Clearly, Excel is not capable of handling complex models (e.g. differential equations). Models created in Excel are sometimes difficult to interpret and cannot be shared across most software packages

2.2. Commercial Mathematical Modeling Packages

It is possible to solve mathematical models in commercial mathematical packages, such as Matlab, Mathematica or Aspen Custom Modeler, so as to utilize the equation solving and visualization capabilities of these tools. This approach provides better usability compared to Excel since the variable names are directly used instead of cell locations or aliases as in Excel. The users who need to use the model however, have to be familiar with the specific syntax designed for that package. Thus, it would be difficult for users other than the developers/experienced users to utilize the knowledge.

2.3. Commercial Flowsheet Simulators

Commercially available flowsheet simulators such as Aspen Plus, Batches provide another approach for mathematical model solving where the model usually exists as a "blackbox" giving the user no freedom to create new models or manipulate existing models. The process of model solving typically is embedded in the form of a flowsheet simulation and the user has drag/drop options to pick operations that have parameters to be solved for. The models behind these operations have been hardcoded by the software developer beforehand.

2.4. Other related work

Bogusch et al. (1997) refer to an ontology based approach for managing process model equations that defines the semantics between the equation and the variables in them. Although many research groups have tried to use such an ontological structured framework for managing mathematical knowledge in process models e.g. ModKit (Bogusch et al., 2001), ProMot (Mangold et al., 2005) etc, most of them have had to rely upon the simulation environment/engine to analyze and solve the systems of equations while powerful dedicated solvers are available.. Marquardt et al (2004) describes design and implementation of ROME (Repository of a Modeling Environment), a model repository to support maintenance of heterogeneous process models and their integration from a data management point of view. The CAPE-OPEN (See URL) standard aims at supporting the modeling process by providing interoperability between modeling tools. While the actual model can exist in any tool by itself, a wrapper has to be written around each model in a markup language called CapeML which creates the input and output ports of the model so that it can interact with other models. While both ROME and CAPE OPEN initiatives try to address the goal of model interoperability and communication, the individual models are still in a format that is specialized and therefore requires familiarity to manipulate models.

3. Ontological approach

In the proposed approach (mentioned briefly in Venkatasubramanian et al., 2006) for modeling mathematical knowledge, the declarative and procedural parts of the mathematical knowledge have been separated. The declarative part consists of the information required by the model to be solved, the information generated from the model, and the model equations. The procedural part consists of the details of model solving such as the algorithms being used and variable initializations. Mathematical Markup Language (See URL) which is based on XML has been used as a standard way of describing the mathematical equations. There are two dialects in MathML: the presentation markup which concentrates on displaying the equations; the content markup concentrates on the semantics (meaning) of the equations.

Among the ontology development tools available for OWL, Protégé 3.2 was selected due to its maturity, ease-of-use, its scalability and extensibility (Castells et al 2004b). The Web Ontology Language (OWL) provides an ontology modeling language with defined formal semantics. There are already tools available both for authoring and parsing various forms of OWL documents as well as tools for reasoning over ontologies expressed in OWL (Obitko et al. 2003). Thus OWL is used to model the ontologies as described below. Fig 1 shows a schematic of the intercommunications in the proposed approach. The declarative part of the approach consists of a *ModelDefinition* ontology (or Model ontology) which contains the definition of the models and an information repository, *Model Use* ontology (or Operation ontology) containing design data such as operating conditions, equipment parameters for the use of models. The model ontology consists of a model class that has as attributes, instances of Equations class that denote the model equations (which in turn can be DAE's, PDE's, Integral eqns, Algebraic eqns or function evaluations) and instances of the assumptions, model parameters, dependent variables, independent variables, universal constants classes. All the above attributes of a model class essentially describe the knowledge about the model in an intuitive and explicit manner which makes the model representation systematic, computer interpretable and generic in nature. One of the biggest strengths of this approach is that the components of the model thus created are entirely reusable i.e. equations, variables, assumptions from one model can be reused while creating another model. Several web based graphical editors are available to create mathematical equations and store them in Content MathML (WebEq, See URL).

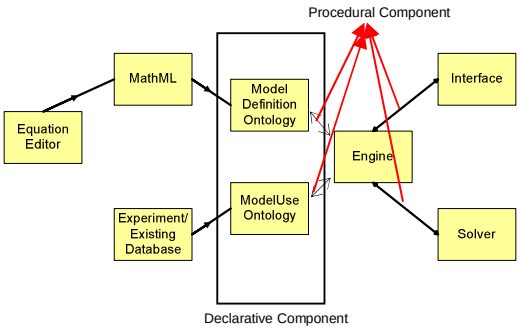

Fig 1. Proposed Approach

Thus, the process of creating a mathematical model becomes very intuitive and user friendly compared to the existing approaches. Each model in this approach is an instance of the model class of the ontology. Model variables are linked to their values using a namespaces recursively which essentially provide the hyperlink to the placeholder of the value in the operation ontology. The *ModelDefinition* ontology also contains the functional representation of the model in the form of Signed Directed Graph (Maurya et al., 2003) for advanced model based fault diagnosis. Fig 2 shows a subset of concepts and relationships captures in the model ontology

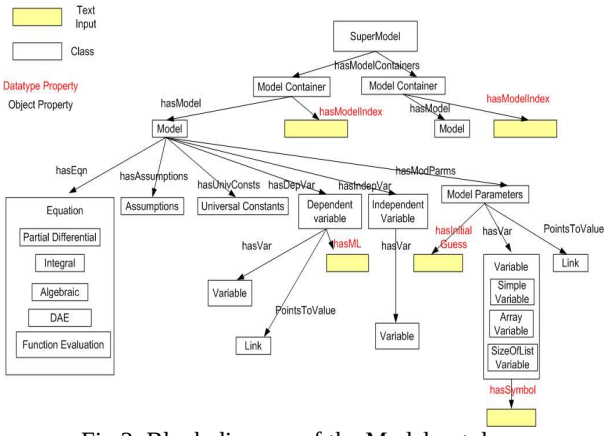

Fig 2: Block diagram of the Model ontology

The *ModelUse* ontology consists of the operation class whose subclasses are the different unit operations in chemical/pharmaceutical product development and instances contain the operating conditions of the operation. This ontology also consists of the results class wherein the link to file containing the results of model solving is stored. The procedural part of this approach consists of a Java based engine for constructing statements that parse the equations in MathML and translating them to expressions interpretable by the Mathematica, which was chosen as the solver. The engine also creates statements to (1) initialize the model parameters with values provided in the instance of the model class chosen, (2) create associations between variable indices and quantities it stands for, (3) initialize universal constants, (4) put together the actual model solving command and (5) invoke the Mathematica kernel to solve the set of equations. Mathematica provides us with many readily useful features, including the symbolic processing capability which handles equations in MathML formats directly and the extensibility with programming languages like Java. A Graphical user interface (GUI) is used to display results from the solver (plots or expressions) along with storing results back to the ModelUse ontology and is also used to select the instance of the model to be solved and the operation to be modeled.

4. Batch Filling model Example

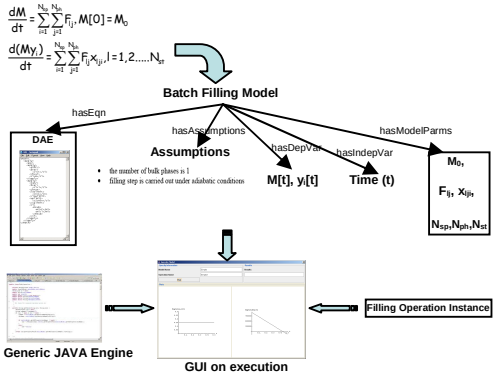

Fig 3. Tank filling model

Fig 3 shows an example of how a batch tank filling operation is modeled using OntoMODEL. The model equations (in this case, mass and species balances) are converted to Content MathML strings and stored in the hasEqn attribute. The assumptions, dependent, independent variables and model parameters are also defined. The JAVA engine on execution lets the user pick the model instance and operation instance after which it constructs

Mathematica statements that construct equations from MathML strings, initialize model parameters based on operation instance, and finally create solve statements that let Mathematica know which variables to solve for. On computation, the results are given back by the Mathematica kernel to the GUI and plots, other results are either shown graphically in the GUI or stored back in the operational instance.

5. OntoMODEL

The resulting tool using the ontological approach described above acts as an 'one-stop shop' for systematic model creating, manipulating, solving, searching and querying which is completely transparent, user friendly and automated. The tool has been shown to work for DAE's, ODE's, and Algebraic equations with unit operation models from pharmaceutical product development, e.g. Johanson's Rolling theory (Johanson et al 1965). The GUI shown in Fig 4 is undergoing constant updates and later versions have features to search and query models, equations etc. Apart from this standalone application, the model repository is also being investigated for use in model predictive control, fault diagnosis and ontology based decision making frameworks.

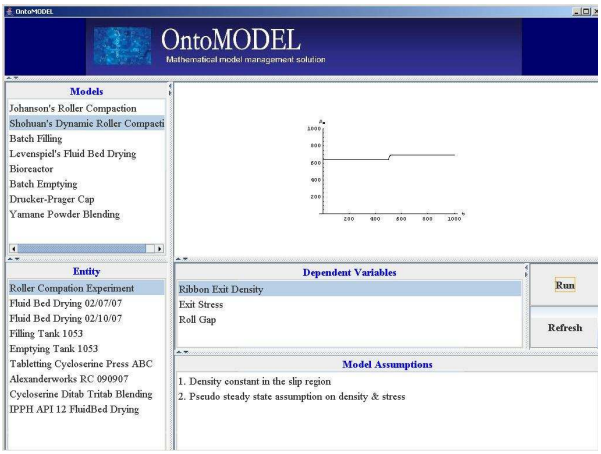

Fig 4. User interface for OntoMODEL

6. Summary

Despite progress in individual fronts of mathematical modeling knowledge management such as sophistication of solvers and standardized markup languages for model representation, progress towards a framework that provides a reusable, user friendly, portable, integrated model management environment was lacking, especially for pharmaceutical product development. In this work, an integrated framework that facilitates the process of mathematical model creation, manipulation, reuse and solution is described. An ontological information-centric approach to model the mathematical information and knowledge is proposed which offers a variety of advantages from the developer and user perspective. Various other applications of this framework, apart from being a useful standalone tool, such as being part of an ontological decision logic framework, model based process operations such as fault diagnosis, control and optimization are being explored.

References

Aspen Custom Modeller, http://www.aspentech.com/product.cfm?ProductID=54

R. Bogusch, W. Marquardt, 1997, A formal representation of process model equations, Computers & Chemical Engineering, 21 (10): 1105-1115

R. Bogusch, B. Lohmann, W. Marquardt, 2001, Computer-aided process modeling with ModKit, Computers & Chemical Engineering, 25 (7-8): 963-995

The CAPE Open Laboratories network, http://www.colan.org/

P. Castells, B. Foncillas, R. Lara, M. Rico and J.L. Alonso, 2004, Semantic Web technologies for economic and financial information management, Lecture Notes in Computer Science 3053 473-487

W. M. Farmer, 2004, MKM: A New Interdisciplinary Field of Research., *ACM SIGSAM Bulletin*, 38(2)

J. R. Johanson, 1965, A rolling theory for granular solids, *Trans. of the ASME: J. Appl. Mech., Series E*, 32(4), 842-848

M. Mangold, O. Angeles-Palacios, M. Ginkel, A. Kremling, R. Waschler, A. Kienle, and E. D. Gilles, 2005, Computer-aided modeling of chemical and biological systems: Methods, tools, and applications, Industrial & Engineering Chemistry Research 44 (8): 2579-2591

M. R. Maurya, R. Raghunathan, and V. Venkatasubramanian, 2003, A Systematic Framework for the Development and Analysis of Signed Digraphs for Chemical Processes. 1.AlgorithmsandAnalysis, Industrial & Engineering Chemistry Research 42, 4789-4810

W. Marquardt, M. Nagl, 2004, Workflow and information centered support of design processes - the IMPROVE perspective, Computers & Chemical Engineering 29 (1): 65-82

MathML, http://www.w3.org/Math/

M. Obitko and V. Marik, 2003, Adding OWL semantics to ontologies used in multi-agent systems for manufacturing, Lecture Notes in Artificial Intelligence 2744 189-200

V. Venkatasubramanian, C. Zhao, G. Joglekar, A. Jain, L. Hailemariam, P. Suresh, P. Akkisetty, K. Morris, G.V. Reklaitis, 2006, Ontological informatics infrastructure for pharmaceutical product development and manufacturing, Computers and Chemical Engineering 30, 1482–1496

WebEq, http://www.dessci.com/en/products/webeq/