Simulation and the Semantic Web

John A. Miller
Computer Science Department
University of Georgia
Athens, GA 30602, U.S.A.

Gregory Baramidze
Computer Science Department
University of Georgia
Athens, GA 30602, U.S.A.

ABSTRACT

One of the missions of the Semantic Web is to put more knowledge on the Web in an organized fashion and link it to other information and data sources. Three successively more capable languages are (or will soon be) provided for this: RDF, OWL, and SWRL. This paper makes a case for using all three for the domain of modeling and simulation. Based on experience developing the Discrete-event Modeling Ontology (DeMO) some observations on the issues and challenges involved in creating such ontologies are presented. An approach for decomposing models into behavioral and observable parts, a la Hidden Markov Models, which can make ontologies smaller and easier to understand, is also discussed.

1 INTRODUCTION

Modeling and Simulation have focused on creating behavioral models that are abstract in the sense that their meaning is primarily in the mind of the analyst. The numbers produced and the graphs generated signify or predict properties of an existing or proposed system. The modern trend toward richer and more realistic animations is certainly a useful way to inject meaning and enhance understandability. This is particularly so as the animations lead toward immersive or game environments. The question is how can we better represent the real world in a meaningful way. Besides better visuals, human understanding is guided by how new results or information are related to existing knowledge. Where does it fit in a semantic space? Although knowledge representation and ontology creation have a long history in Artificial Intelligence and have impacted simulation in the past in the form Knowledge-Based Simulation (Fox et al., 1989), the new impetus to create a Semantic Web (Berners-Lee et al., 2001) should provide enough leverage to make adding semantics useful for the modeling and simulation community.

Current research and development into what is called the next generation Web, or the Semantic Web promises to transform the Web by providing machine-processable and meaningful descriptions of Web resources. This can improve discovery, integration and use/reuse of Web resources, and we believe it also holds significant promise for the Simulation and Modeling community.

Much of this work involves the use of ontology (Gruber, 1995) to define terms or concepts in certain domains. For a particular domain, types of things are defined as classes, which have properties and relationships. The meaning of a concept is captured via the concept's/class's position within a taxonomy (subclass-of/is-a hierarchy) as well as its properties, relationships and restrictions.

Many fields have recently been creating ontologies and making them available over the Web. In the biological domain, several ontologies have been created. Creating ontologies for modeling and simulation is harder for two reasons: (1) It is not domain limited since models may simulate biological, chemical, physical, clerical, transportation, military, services or manufacturing, etc. (2) Modeling and simulation methodology is founded in mathematics, probability and statistics and hence to be rigorous about it, ontologies for these fields should serve as foundations (as so called mid-level ontologies). The mid-level ontologies should themselves be ideally based on broad, standardized upper ontologies such as SUMO or SUO.

In this paper, we will address these obstacles, review the DeMO ontology including recent additions and discuss the benefits of such ontologies. In section 2, we discuss relevant aspects of the Semantic Web. A case for using Semantic Web technologies for modeling and simulation is made in section 3. Section 4 gives an overview of the DeMO ontology and brings up some issues in its creation. Section 5 discusses several benefits of having more semantics available from the point of view of (1) discovering and (2) using resources effectively. An approach for adding observable models as a new part of DeMO is considered in section 6. Finally, section 7 gives conclusions and discusses future work.

2 ONTOLOGY AND THE SEMANTIC WEB

The World-Wide is undergoing a slow transformation from HTML (statically or dynamically created) to XML and XML-based languages. XML adds more structure and more meaningful tags to documents on the Web. Furthermore, there are schema languages for specifying the structure for valid documents (e.g., DTD, XML Schema or Relax-NG). In addition, there are other XML-based files being place on the Web to describe and link ordinary Web documents. These can be thought of as meta-data, or data about the data that is on the Web. Predominantly, the Resource Description Framework (RDF) is used to hold this meta-data. RDF has a simple approach to capture this information as subject-predicate-object triples, which are akin to simple English sentences. The subjects and objects may be treated as nodes and the predicates as edges, so that RDF data may be conceptualized as or stored as a labeled directed graph. The labels for nodes and edges come from classes and properties defined in an RDF Schema, respectively.

An RDF Schema may provide a simple domain model. However, the current approach is to use a language that is richer than RDF Schema and simpler than traditional ontology languages such as The Knowledge Interchange Format (KIF). The reasons are that RDF/RDF Schema may be too spartan for effective domain modeling and not ideally suited for automated reasoning (e.g., logical entailment). Conversely, some believe that the use of KIF at a Web scale is not feasible. Consequently, the Web Ontology Language (OWL) was developed. Recognizing the expressivity-complexity tradeoff, OWL comes in three flavors: OWL Lite, OWL DL and OWL Full, which fall into the following complexity classes for automated reasoning (e.g., subsumption and satisfiability), EXPTIME, NEXPTIME and Semi-decidable, respectively. OWL is based on Description Logic (DL) and supports several useful constructs for defining classes, properties (data type properties or object properties) and restrictions.

Description Logic is a subset of First Order Logic, but lacks the ability to define rules as one might see in Expert Systems or Prolog. Since rules are an effective way to encode useful knowledge, there is a need to introduce such a language. The proposed Semantic Web Rule Language (SWRL) represents such an effort. Presently, rules in SWRL are made up binary Horn clauses. There is ongoing research to both weaken (for the sake of expressivity) and strengthen (for the sake of complexity/computability) the restrictions. Perhaps, the multi-language approach taken by OWL is the best way to handle these conflicting goals. The long term plan for the Semantic Web as depicted in its Layer Cake Model (*www.w3.org/2004/Talks/0412-RDF-functions/slide4-0.html*) includes layers above the SWRL rules/logic layer which might include Higher Order Logic.

Finding information in the new Semantic Web will likely become some hybrid of information retrieval, navigation and query processing. Along with the Semantic Web languages, corresponding query languages are being developed: RQL, RDQL, nRQL, OWL-QL, etc. As the query languages become more powerful, the complexity and query processing times go up. In (Zhang and Miller, 2005), it was found that RDF based querying scales well as the size increases, while OWL based and SWRL based querying does not. It is still early in the R&D cycle, so progress on these may be expected in the future. As an example of the complexity, the paper uses the Vampire First Order Logic Theorem Prover to process SWRL queries. Since the problem is semi-decidable, the theorem prover must be given a time limit, as there is no upper bound on the running time.

3 ADDING SEMANTICS TO SIMULATION

A skeptic might claim that modeling and simulation are general purpose techniques that achieve their usefulness through abstraction. In this way a bank and drive through restaurant can be modeled in similar ways using abstract queues with different parameter values for interarrival time and service time. It is great to be able to abstract out the essential features and discover fundamental similarities. The modeling and simulation community has been doing this successfully for decades. In our opinion, this paradigm has three weakness: (1) The mapping from the real world to the abstract model is largely in the mind of the simulation analyst. (2) High fidelity, multifaceted modeling is difficult to achieve. (3) Building models out of model components is limited.

This paper hypotheses that adding semantics to modeling and simulation will to some degree overcome these weaknesses. Providing information about what things mean is needed. This is done today using natural language, yet to achieve the advantages of being machine processable and searchable, a formal language should be used. This is where the Semantic Web comes in. If the meaning of models, model components, simulation results, simulation studies, etc. can be given using a Semantic Web language, then machine processing (including logical inferencing) and effective search can be preformed on a Web scale.

Many scientific domains have been developing and using ontologies. Biological sciences have demonstrated the greatest success as can be evidenced by looking at the Open Biomedical Ontologies (OBO) site (Open Biological Ontologies, 2005), that lists 29 ontologies including the most famous Gene Ontology (GO). Developing ontologies for modeling and simulation presents a challenge in that it is built on mathematics and statistics, so that if depth and rigor are desired, more powerful languages are needed. One approach is to define a minimal set of mathematical and statistical terms (e.g., set, function, probability, etc.) in natural language and build the rest of the ontology on top of these. This was the approach taken in the DeMO ontology (Miller et al., 2004).

As the Semantic Web begins to provide more expressive languages, it becomes of more use to the Simulation and Modeling Community. For example, more complex concepts can be defined using SWRL. As of the Summer of 2005, the leading ontology editor, Protege, will support many of SWRL's built-in operations, such as $lessThan$. In SWRL, properties of probability, such as sub-addivity, can be stated as follows:

$$lessThan(P(union(A, B)), add(P(A), P(B)))$$

The Mathematics on the Net (MONET) project (*monet.nag.co.uk*) is making progress toward putting mathematics on the Semantic Web. Much of this work involves translating the OpenMath (*www.openmath.org*) Content Dictionaries (CDs) into a Semantic Web language, OWL for now. There exist CDs for Calculus, Combinatorics, Differential Equations, Geometry, Linear Algebra, Logic, Permutations, Physical Constants, Polynomials, Sets, Special Functions, Transcendental Functions and Units & Dimensions. This work will also include translations to common markup languages used for mathematics such as MathML (*www.w3c.org/Math*) and LaTex.

4 OVERVIEW OF THE DeMO ONTOLOGY

Work on the Discrete-Event Modeling Ontology (DeMO) began in 2003 (Miller et al., 2004; Fishwick and Miller, 2004) to explore issues and challenges in developing ontologies for simulation and modeling. As its name suggests, it is focused on discrete events models. It used the OWL language to define over 60 classes and many properties. Figure 1 in the Appendix is a screenshot from OWLViz showing the DeMO class hierarchy. The ontology consists of four main parts: `ModelConcept`, `DeModel`, `ModelComponent` and `ModelMechanism`. `DeModel` is itself divided into four parts based on the three simulation world views plus a fourth representing state models, namely, `StateOrientedModel`, `ActivityOrientedModel`, `EventOrientedModel` and `ProcessOrientedModel`. Note, the screenshot shows DeMO version 1.8, which is missing the `ProcessOrientedModel` subtree, which is going into version 1.9.

As illustrated by the OBO site, it is better to have several (but not too many) interrelated ontologies, rather than one huge monolithic ontology. Along these lines, DeMO as it is extended, could be divided into more than one ontology. In addition, DeMO ignores much of the simulation domain such as continuous models, statistical modeling, output analysis, random variates, etc. Also, DeMO at present has few instances. One could attempt to populate the ontology (or knowledge-base) with information about simulation engines, available simulation models, model components, etc. This could be done by writing extractors for scanning the Web for information or by providing a mechanism for publication. Alternatively, one could simply use the ontologies for annotation of simulation artifacts (as is done in the proposed WSDL-S (Akkiraju et al., 2005) standard). Then special semantic search engines could precisely retrieve the information requested.

4.1 Strategies for Creating DeMO

Models may or may not be used for running a simulation. Simulation may or may not require input and output functions. There can be several ways to approach the descriptions of different modeling formalisms (formalism specification).

One way is to consider each model separately and define them from scratch. This may be called a "problem in hand" approach - given a problem, define a modeling formalism that "fits" the problem well. Such an approach is by far the most popular one and explains a great assortment of modeling formalisms existing in scientific literature and used in practice by the simulation community. This is also a most natural approach from a practical point of view: different modeling formalisms "fit" differently into different problems; some are more fitting for one purpose, some for another. The user is usually not particularly interested in how a model she/he employs is related to other modeling formalisms as long as it satisfies her/his design goals.

Another way is to define some very general formalism and consider all other models to be some sort of subformalisms - restrictions on a general framework such as DEVS. This view is logical and natural as well, because most of the existing modeling approaches easily subject themselves to formal description and it is only a question of finding a general enough framework that encompasses all the existing formalisms and from which new sub-formalisms can be derived.

DeMO utilizes a middle ground approach: several general (upper-level) formalisms are defined independently of each other. (Of course they do not have to be completely independent of each other and may themselves be derived from some even more general formal framework.) These upper-level formalisms can be viewed as root classes for a taxonomic tree for the discrete-event modeling and simulation domain. All other modeling formalisms are defined as restrictions on one of the root classes.

Importantly, DeMO uses a uniform approach to a description process of a modeling formalism. Each DeModel is considered as having Model Components and Model Mechanisms (syntax and semantics of the model), which in turn are defined using fundamental Model Concepts. This approach allows for great flexibility and straightforwardness in constructing an ontology and defining new formalisms. Another (and most important) unifying feature of DeMO is that it represents knowledge using an XML-based, Web-oriented language (OWL) which opens up an abundance of possibilities for its use in the field of Web-based simulation and modeling.

5 APPLICATIONS AND BENEFITS

Ontologies like DeMO ultimately should be judged by the benefits they bring. We envisage several possible benefits or usages for such ontologies. Many have been listed in the other DeMO papers (Miller et al., 2004; Fishwick and Miller, 2004). In this paper, we highlight three related to discovery and three related to usage.

- Browsing. Using an ontology editor, such as Protege, one can browse through the ontology looking for classes or properties. One could look for certain concepts in the ontology and navigate to related concepts. This easily can be done with Protege by expanding/contracting the class hierarchy tree.

- Querying. As ontologies become large, browsing can become tedious. For ontologies with a huge number of instances (tens of thousands), browsing is nearly infeasible. Query languages have been developed by the database community for just this purpose, conveniently accessing large amounts of data. In addition, the information retrieval community has been developing search engines for years that utilize keyword lookup among other things. Query languages for ontologies or the Semantic Web in general combine elements of each. The paper by (Zhang and Miller, 2005) provides an evaluation of several of these languages.

- Visualization. Browsing is useful when the ontology is small or for localized access. Querying is good, when you have a good idea of what you want. Suppose you would like an overall view of the ontology or parts of it as a starting point for browsing. This is where visualization of ontologies comes in. Since the schema part of an ontology (class and properties) is usually not extremely large, viewers that display the ontology as a zoomable and panable graph can be effective. Currently, there are several viz tools that work as Protege plugins, including EzOWL, Jambalaya, OntoViz, TGVizTab and OWLViz.

- Components. The three items mentioned above allow elements (classes, properties and instances) to be found. In some cases, the retrieval of this information or knowledge is an end in itself. In other cases, it is the first step in a simulation study that includes model building, scenario creation, model execution, output analysis and saving/interpreting results. Ontologies can be useful during all of these phases. During model building, one could search for appropriate simulation engines to execute models. If using an extensible engine (e.g., support customization, plugins or a service oriented architecture), then components can be found and added in. For example, code implementations of random variate generators for specific probability distribution functions may be linked to an ontology.

- Multi-modeling. At a more coarse level of granularity, models relevant to the system under study may be found. One or more of these models may be run to generate results. In some cases, the ever increasing complexity of modeling and simulation problems gives rise to the need to combine multiple models. Multi-modeling and meta-modeling (Vangheluwe et al., 2002) techniques allow different models as well as different model formalisms to be coupled or combined together to model complex systems. An existence of a unifying ontology can simplify both the design of the multi-models and their implementation.

- Multi-faceted Reasoning. Beyond modeling building, their is the issue of generating and interpreting results. Possibilities exists for combining inductive and deductive reasoning with simulations generating data, statistics and data mining adding rules which interact with the logic and rules given in OWL and SWRL.

6 OBSERVABLE MODELS

A recent direction taken in the development of the DeMO ontology is decoupling the model dynamics (what is really happening as the model executes over time) from how it is observed. Since many models can be observed in many ways, this decoupling has the potential to reduce the size of the ontology with no loss of information. Simple examples of this are the Mealy and Moore machines for observing the outputs of Finite State Machines. In some cases, observable parameters can be measured, while the underlying parameters driving the behavior are hidden and can only be indirectly estimated.

In DeMO, one may think of models/formalisms as entities disconnected from the observer. In other words, DeMO defines how the model can run, but does not say anything about what it outputs. We consider the output function as a separate entity. It is not a fixed part of the model, but rather an observation device that can be applied to the model. In other words, a given model/formalism can be observed through the actions of different output functions that fit this formalism.

DeMO defines a model as a set of components and mechanisms (its syntax and semantics). This is enough to construct a simulation kernel for the model capable of running a simulation. We can interpret the simulation as observing the state-trajectory of the model (in this case a sequence of states, but more generally this includes all parameters of the model: states, which events were activated and when, which transitions were triggered and when and so on). That is, we assume that the observer can view all aspects of model behavior and moreover he/she can view it directly (perfect translation).

In a more general setting, however, we may want to restrict the observation window of our formalism to fit certain aspects of a modeled system. In the case of the Markov Chain formalism the observer, for example, may not be able to directly observe the states as the simulation runs. Instead she/he may only be able to see an imperfect translation of the sequence of states. For instance, each state may with a certain probability emit some symbol that the observer can see.

This is the case in Hidden Markov Models (Rabiner, 1989), a popular statistical model used in many different fields from bioinformatics to linguistics. The model has hidden states $S = s_1, s_2, ..., s_n$ and observable outputs $V = v_1, v_2, ..., v_m$. It is based on the Markov chain in that the state transition is a stochastic function of the current state. This means that the next state is only dependent on the preceding state, not on any past states. In a sequence of hidden states, each state q_i takes a value from S. Each output o_i in the observable sequence takes a value from V.

This model is usually interpreted as a Markov model with unknown parameters. Generally speaking, this can be handled in several different ways. One, for example, is to add properties observable and hidden to states in Markov Chain formalism and define an Emission Function concept in addition to a Transition Function. Alternatively, one can consider a Hidden Markov Model to be defined as a restriction on the Semi-Markov Process class, where we have two types of events: transition events (that result in hidden state transitions) and emission events (these are only allowed to result in hidden state to observable state transitions).

We, however, prefer another approach. To give any meaning to this formalism we must attach a notion of output function to DeMO. We, therefore, view it as the Markov Chain with a stochastic output function. More precisely, the output function is defined as probabilistically mapping a current state of the system to one of the output symbols. In general, an output function needs to define its domain (normally one or more of Model Components) and range (usually a finite set of symbols, an output alphabet) as well as the mapping rules. We think that it is more advantageous to define an output function in a separate part of the DeMO ontology, emphasizing the fact that it is not inherently part of a model, but more of an attachment to the model. This fifth part of the DeMO ontology is to be included in the next release, DeMO 1.9.

We believe that there are good reasons to follow such an approach. This separation of models and their (output) interfaces provides for greater flexibility and modularity in defining observable models. Indeed, given a model defined in DeMO, we can attach different output functions to it to produce new observable models. Moreover, different model formalisms may use the same output functions (if they fit properly). The alternative would have been defining each of these formalisms separately in DeMO, thus, substantially increasing the size and complexity of the ontology.

Conceptually as well, models (modeling formalisms) may be viewed to be independent of the results of simulating these models. It is hoped, however, that a well designed ontology will allow capturing of different points of view on the knowledge domain without explicitly categorizing each one of them. For example, it is hoped that an automatic reasoner should be able to deduce that a Hidden Markov Model defined as above is in essence equivalent to a restricted Semi-Markov model with direct translation.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we have looked at reasons that emerging Semantic Web technology can be useful in modeling and simulation. Although not as natural a candidate as the Biological Sciences, recent developments, such as OWL and SWRL and the ability to use them in ontology editors such as Protege, make the time ripe for the development of ontologies for modeling and simulation. Experience in developing the DeMO ontology has lead to some fundamental questions that are difficult to solve. There are many modeling formalisms and many of them are similar, so what is the best way to make a hierarchy of model classes (one very general model class as root, several roots or the approach taken by us of picking four based on simulation world views)? What steps can be taken to reduce the size of the ontology without (1) losing knowledge or (2) making the knowledge appear convoluted (or difficult) to understand?

Future work includes the following: (1) Since in the Summer of 2005, Protege will support built-in operations in SWRL, adding rules to DeMO would be useful. (2) Greater understanding of the similarities and differences between the model formalisms within the DeModel hierarchy, can be achieved by providing morphisms. It is worthwhile to investigate whether this can be done within the ontology itself, for example, using SWRL. (3) Populate the ontology and develop more specific usage scenarios.

REFERENCES

(Akkiraju et al., 2005) R. Akkiraju, J. Farell, J.A. Miller, M. Nagarajan, A.P. Sheth and K. Verma, "Web Service Semantics - WSDL-S," *Proceedings of the W3C Workshop on Frameworks for Semantics in Web* Service (W3CW'05), Innsbruck, Austria (June 2005).

(Berners-Lee et al., 2001) T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web," *Scientific American*, Vol. 284, No. 5, pp. 34-43.

(Fishwick and Miller, 2004) P.A. Fishwick and J.A. Miller, "Ontologies for Modeling and Simulation: Issues and Approaches," Proceedings of the 2004 Winter Simulation Conference (WSC'04), Washington, DC (December 2004) pp. 259-264.

(Fox et al., 1989) M.S. Fox, N. Husain, M.McRoberts, Y.V. Reddy "Knowledge Based Simulation: An Artificial Intelligence Approach to System Modeling and Automating the Simulation," *Artificial Intelligence, Simulation and Modeling*, L.E. Widman, K.A. Loparo, and N.R. Nielsen (Eds.), Wiley Interscience, N.Y.

(Gruber, 1995) T.R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Int. Journal of Human-Computer Studies*, Vol. 43, pp. 907-928.

(Miller et al., 2004) J.A. Miller, G. Baramidze, P.A. Fishwick and A.P. Sheth, "Investigating Ontologies for Simulation Modeling," *Proceedings of the 37th Annual Simulation Symposium*, Arlington, VA, April 2004, pp. 55-71.

(Vangheluwe et al., 2002) H. Vangheluwe, J. Lara, P.J. Mosterman, "An Introduction to Multi-Paradigm Modelling and Simulation," *Proceedings of AI Simulation and Planning (AIS-2002)*, Lisbon, April 2002, pp. 9-20.

(Rabiner, 1989) L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-287.

(Open Biological Ontologies, 2005) Open Biological Ontologies Project, "OBO Ontologies," Available online via http://obo.sourceforge.net/cgi-bin/table.cgi [accessed June 28, 2005]

(Zhang and Miller, 2005) Z. Zhang and J.A. Miller, "Ontology Query Languages for the Semantic Web: A Performance Evaluation," *Journal of Web Semantics.* (under review)

AUTHOR BIOGRAPHIES

John A. Miller is a Professor of Computer Science at the University of Georgia and has also been the Graduate Coordinator for the department for 9 years. His research interests include database systems, simulation, bioinformatics and Web services. Dr. Miller received the B.S. degree in Applied Mathematics from Northwestern University in 1980 and the M.S. and Ph.D. in Information and Computer Science from the Georgia Institute of Technology in 1982 and 1986, respectively. During his undergraduate education, he worked as a programmer at the Princeton Plasma Physics Laboratory. Dr. Miller is the author of over 100 technical papers in the areas of database, simulation, bioinformatics and Web services. He is an Associate Editor for ACM Transactions on Modeling and Computer Simulation and IEEE Transactions on Systems, Man and Cybernetics as well as a Guest Editor for the International Journal in Computer Simulation and IEEE Potentials.

Gregory Baramidze is a Ph.D. student in the Computer Science Department at the University of Georgia. He received the M.S. Degree in Applied Mathematics from the University of Georgia in 2001. His research interests include Theoretical Computer Science, Quantum Computing and Modeling & Simulation.

APPENDIX

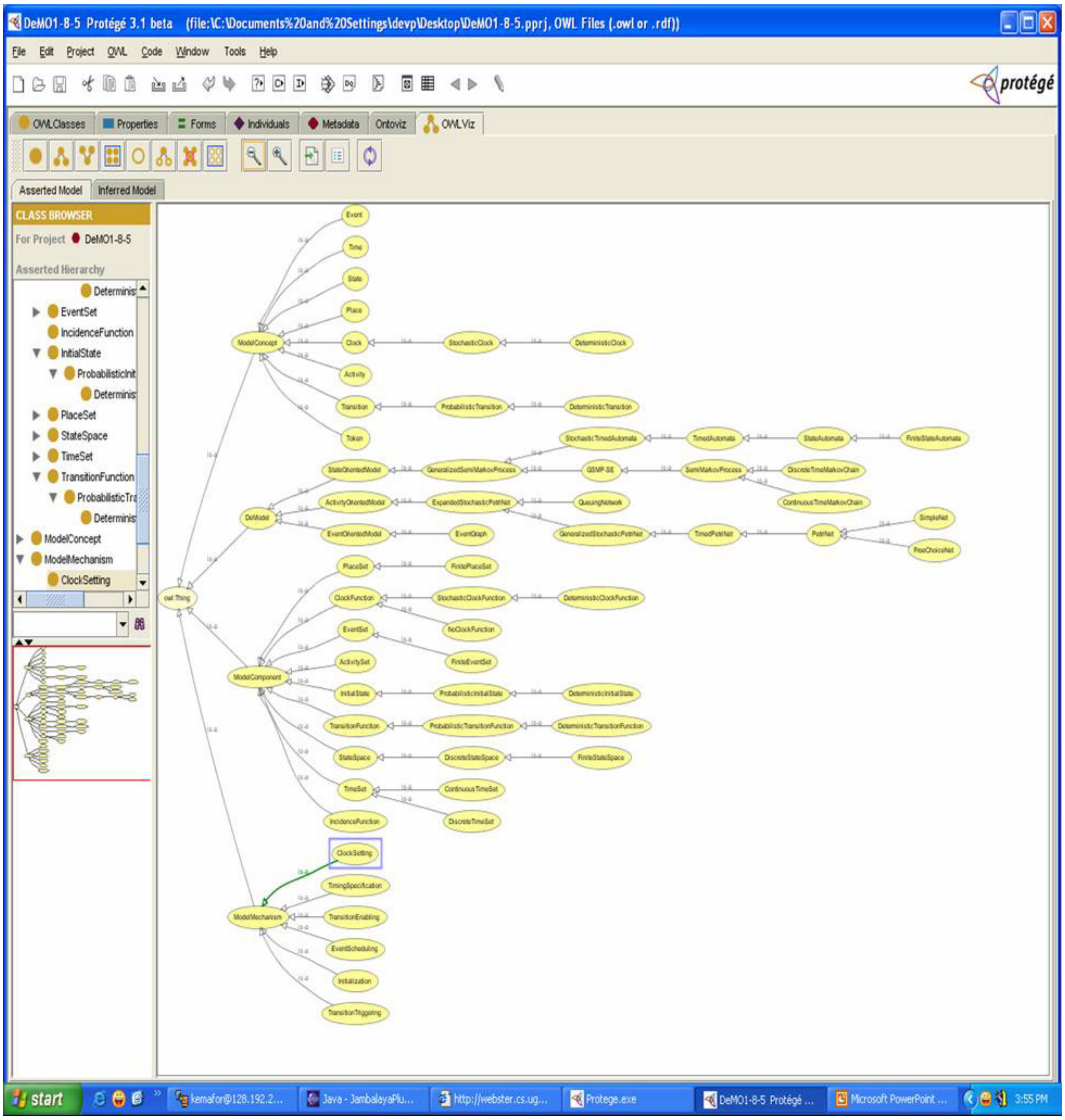

Figure 1: DeMO Class Hierarchy shown using OWLViz