Issues with Web-Publishable Digital Objects

P. A. Fishwick

Computer and Information Science and Engineering Department
University of Florida, Gainesville, FL 32611

ABSTRACT

Our goal is to promote the publication and standardization of *digital objects* stored on the web to enable model and object reuse. A digital object is an electronic representation of a physical object, including models of the object. There remain many challenges and issues regarding the representation and utilization of these objects, since it is not clear, for example, who will create certain objects, who will maintain the objects, and what levels of abstraction will be used in each object. This article introduces some of the technical and philosophical issues regarding digital object publication, with the aim of enumerating technical, sociological and financial problems to be addressed.

Keywords: Digital Object, Multimodeling, Model Abstraction, Object Orientation

1. INTRODUCTION

One of the most critical problems in the field of computer simulation today is the lack of published models and physical objects within a medium—such as the World Wide Web—allowing such distribution. The web represents the future of information sharing and exchange, and yet it is used primarily for the publication of documents since the web adopts a "document/desktop metaphor" for knowledge. In the near future, we envision an "object metaphor" where a document is one type of object. A web predicated on digital objects is much more flexible and requires a knowledge in how to model physical phenomena at many different scales in space-time.

If a scientist or engineer (i.e., model author) works on a model, places the model inside objects, and constructs a working simulation, this work occurs most often within a vacuum. Consider a scenario involving an internal combustion engine in an automobile, where the engine is the physical object to be simulated. The model author's task is to simulate the engine given that a new engineering method, involving a change in fuel injection for example, is to be tested. By testing the digital engine and fuel injection system using simulation, the author can determine the potential shortfalls and benefits of the new technique. This task is a worthwhile one for simulation, and simulation as a field has demonstrated its utility for objects such as engines.

Let's analyze the problems inherent in this example. There is no particular location that will help the author to create the geometry of the engine and its dynamics. It may be that other employees of the company have made similar engine models in the past, and that these models may be partially reused. If this is the case, the model author is fortunate, but even if such a company-internal model exists, it may not be represented in "model form" (ref. Sec 3). There may be other model authors who have already constructed pieces that our model author could use, but there if there is no reuse and no standard mechanism for publishing the model or engine object, then this is all for naught. The model author may also be concerned with creating a fast simulation. While algorithms for speeding simulations are important, by solving the reusability problem, we also partially solve the speed problem since published quality models of engines will battle in the marketplace for digital parts, and the best engine models and testing environments–involving very fast and efficient simulation algorithms–will win out in the end. Therefore, the problem of reuse of engine objects and components lies at the heart of the simulationist's dilemma. Fast, efficient and quality models could be available at some point in the future, but today there is no infrastructure or agreed-upon standards (ref. Sec. 11) for true digital object engineering.

Many of the issues surrounding digital objects and their representation can be resolved, at least partially, using the *physical metaphor*. We ask a question such as *Who will maintain a digital object?* or *Where is the digital object located?* and we obtain answers by phrasing the question within the corresponding physical domain, yielding *Who maintains the physical object?* and *Where are the physical objects created?* The answers to the latter question

Other author information:Email: fishwick@cise.ufl.edu; Telephone: 352-392-1414; Fax: 352-392-1414; Supported by the U.S. Air Force, GRCI, Incorporated, and by the ATLSS Project of the Department of the Interior.

suggests possible answers to the former. This is a simple technique, but fairly powerful in addressing many of the issues that we will present. We will proceed to outline problems involving digital objects, first by defining them and then continuing with issues that surround digital objects and their future. We hope that this paper can serve as a starting point to debate some of these issues. Some issues were addressed in detail at a recent innaugural conference on web-based modeling and simulation[1*].

2. DIGITAL OBJECTS

Before enumerating the problems that will face the model author, we will state our goal, which is to provide a representation for digital objects on the web. A "digital object" is the digital counterpart to the physical object. Therefore, the digital object contains attributes and methods, where some of these will be models. Two prominent models are the geometry model and the dynamic model since these models capture the shape and behavior of an object. For the engine, we might publish a geometry model based on NURBS (Non-Uniform Rational B-Splines), and for the dynamics, we might create a set of equations representing the transportation of gasoline through the pipelines involved in fuel-injection. These two models are components of the specific engine object and could also be components in an Engine class from which an engine object is created.

Who will initiate the process of storing digital objects? Where is the incentive? This is less of a technical consideration and more a question of whether or not we should deliberate on digital objects. In the remainder of the paper, we'll present arguments for the deliberation issue. In terms of incentive to create digital objects, the marketplace will play a key role. Those industries that plan to sell their products will find their sales increased if they offer their customers an opportunity to test digital products prior to buying the real ones. Currently, some companies on the web offer pictorial and technical specifications for their products, but these are not a complete substitute for digital objects. We feel that the federal government will play a major role in the adoption of standards and digital object proliferation. In particular, during DoD acquisition phases, a stipulation can be put in place where vendors must deliver bids with an accompanying digital object specification.

For the remainder of the article we will often refer to different components such as models, objects and classes. Objects can theoretically be defined without an associated class; however, it is most useful to create a corresponding class when creating any object. This fosters reuse and inheritance and makes it possible to create similar objects from the class. Models reside inside classes and objects.

3. MODEL VERSUS CODE

In many cases, the model (defined as an abstract, and often visual, representation of the engine's behavior), may not be surfaced at all. Instead, there may be a large program whose simulation yields the object's behavior. The model author for the engine, once coding is complete, may speak of having generated a "model", when in reality there is no model except in the model author's mind or sketched on a notepad. The cognitive, notepad-style, model is what is required to be surfaced where it serves as the human-computer interface. In this sense, a "model" and a "human-computer interface" are inseparable. The model serves as the interface which is compiled into target code with which the author need not be concerned. Unfortunately, program code is not a good substitute for a model even though it could potentially be seen as one in the extreme. The "model" must be abstract and must represent a close match to the model author's cognitive map of the engine. Programs tend to obscure these maps by focusing on detailed semantics rather than a more abstract syntax. Most model authors tend to like visual representations of phenomena, and the author of the engine model is no exception. Some of the models that the author will use will be equational in representation, and other models will be graphical.

In addition to graphically oriented cognitive representations that require surfacing, it is also critically important to have an underlying formal semantics for each model that can be accessed to resolve ambiguities and disputes. The more abstract the representation, the more easily it can fall victim to issues involving semantics. So, it is necesssary to maintain a chain of translation from high level abstract model to the semantics that are defined to a level of detail so that execution of the model is unique and unambiguous.

*An online version of these proceedings is available through the web page `http://www.cise.ufl.edu/~fishwick/webconf.html`.

4. REUSABILITY AND SPEED

The author for the engine may find it frustrating to have to reinvent the wheel by constructing a new model. This is where the expense of simulation raises its head. Simulation, as a method, is expensive because of the lack of available digital objects and models that can be reused. While it is true that this particular author may have a new set of questions to be asked of the model, the author could benefit from reuse with modification of the model to fit the author's specific questions to be answered from the simulation.

There is much research into the problems of speeding up model executions. The engine author may well wait hours for results from the simulation. This is unfortunate and costly in terms of hours and machine time. As a simulation community, we need fast algorithms, but without well-published digital objects and their component models, the speed issue resolves problems for only one project and one model author. With published digital objects, representing test environments for fast simulation, authors can spend less time worrying about speed and more time focusing on model structure and design. The web-based marketplace of objects will be fundamental in solving the speed problem just as the physical marketplaces does for the survival of only the most efficient objects.

What is to be reused? Certainly, objects will be reused just as physical objects are reused in the form of "plug and play" techiques used to construct everything from engines to houses. However, the models inside the objects should be reused and the classes, from which many objects will be created, serve as a vehicle for reusability. Therefore, reusable components are *class*, *object* and object structure in the form of individual *models*.

5. INTERFACE AND COUPLING

How might we mix and match components for an engine? Since the fuel injection delivery pipelines physically meet the gasoline tank, the digital equivalents must perform a similar conjunction. The interface to a digital object must be specified to ensure that the data types, at the very least, match. We let an object have three types of ports: input, output, and input/output. Objects can connect to one another via these ports. A line that extends from port 3 on Object A to port 1 on Object B must carry the same type of signal, and the data structure associated with the output on port 3/Object A must match the input on port 1/Object B. There may be additional constraints that can be specified for minimizing problems. These constraints can be specified in a formal constraint language or the language of predicate calculus.

6. AUTHORING AND MAINTENANCE

Who will create the digital objects? Should there be one repository for a specific type of object or should we compete in the marketplace of objects with multiple authors? A reasonable strategy is to let the marketplace dictate which authors are most successful. Some authors may be interested in their own particular object and internal models, whereas others may have alternate motives (ref. Sec. 9). There are many potential strategies for locating digital objects. One strategy suggests that digital objects be located where their physical object counterparts are manufactured. Therefore, the author the company creating the automobile engine will create the digital automobile engine, and the company making the piston rings will create the digital piston rings. We will term this strategy *developer-based colocation*. Reuse can be created at every level so that even the engine author can reuse objects since it is doubtful that all engine subcomponents are manufactured by the automobile company. While the developer-based strategy is appropriate for engineered objects, we must concern ourselves with natural phenomena as well. What about a representation of the Everglades National Park for ecological studies? The *management-based colocation* strategy suggests that the Department of the Interior, which is responsible for this piece of land and its ecology, host the site where the digital Everglades models[2] can be easily accessed. The Department of the Interior can likewise create contracts for industries to compete for the right to carry the objects on their sites, or for a more generally accessible site controlled by the Department. A more arbitrary approach suggests that digital objects can be located physically anywhere on the web. Experts in automotive design at a University might host a site containing digital engines or subcomponents.

7. MODEL ABSTRACTION, COMPLETENESS AND REUSE

Let's say that our model author finds an engine that contains fuel injection geometry and dynamics. Is this particular object appropriate for the model author's simulation requirements and goals? This is a particularly acute problem and one that is central to many issues concerning digital objects. First, we must acknowledge that the author may

indeed find that the engine that he obtains on the web may answer a certain percentage of the questions he wants answered, say 75%. Second, another engine may be available that is different from the first model, and yet contains some extra model semantics which will allow for answering another 20% of the remaining questions. At first, it may appear impossible to provide the functionality in a digital object that will satisfy everyone. This is true. No single object will satisfy everyone, however, there are key considerations and steps toward meaningful digital objects. First, we note that objects should be created, as in the physical world, with the ability to accept input while not dictating the exact nature of that input. Finite element programs and programs based on Newtonian physics are able to simulate a very large class of system. This is because, for instance, forces that affect an object may come from an infinite number of sources, but the source identity is not relevant if the object's input is a force or a vector field of forces. The manifold for an automobile engine may be affected by a wide variety of physical objects, but the identity of these objects does not affect the physics since one is concerned with an impressed, generalized force. The manifold is unconcerned whether the applied force eminates from a person's hand, a wrench or gas expansion. If this invariance to input did not exist, today's engineering software would be severely limited.

Through multiple inheritance, it is possible to inherit all necessary components to create a new hybrid class. The model author, once finished with the simulation, may decide to publish this hybrid class, thereby augmenting the base set of digital objects on the web. Dynamic models and methods incorporating finite element calculations can be combined with point-mass calculations through multiple inheritance, and need not necessarily be located in one monolithic class structure. Also, some of the models required may be located in fundamentally different objects that have the similar methods to what is required. The broader the dynamic or geometric model, the greater the possibility for the author to locate models instead of just objects that match the existing requirements. The author may also find that he needs to create some new models that he cannot locate either in class, object or model form. At the very least, the author has minimized wheel reinvention through a comprehensive search of the web for classes, objects and components. There is also the option of accepting a component that answers all the critical questions to the required level. There are conscious tradeoffs to be considered. We make these same tradeoffs when buying physical objects that may solve most of our needs but not all of them.

Most objects will contain multiple abstraction levels. The abstraction levels presented may not exactly match what is required, but through a search process and through reuse, we can create new levels if they are needed. It is probably unrealistic to imagine that published digital objects will behave in every way, and at every abstraction level, that the physical objects behave. This suggests that model authors who do publish objects be careful about stating up front the constraints of their objects–what they can do and what their limitations are. Over time, and given a free market for digital objects, we estimate that objects will improve over time to yield better and more accurate results that appeal to greater numbers of model authors. Moreover, the taxonomy of objects in the form of class hierarchies will improve in structure to maximize the benefits of aggregation and inheritance.

8. ACCESS

Given that the author of the engine needs to find objects and components, how is he to locate and access them? The most straightforward idea is that model search can be seen as similar to text-based search in modern web search engines. To find an engine, search for the keywords "automobile engine." The result should be a conceptual model consisting of classes and relations. The class from which a specific object is created can be highlighted and displayed with its immediate class context. This is similar in concept to the way that *Yahoo!*[†] organizes its subject taxonomy, except that our tree is concept/class-based. Other search methods include picture-based search and an immersive 3D-based search for objects.

Model repositories will contain class hierarchies and embedded geometry and dynamic models, and we envision that model repositories will proliferate over the web to support the model author. Along with the need for repositories, will be the need for *model bases* to support concurrent access, protected model information, querying and caching of model structure. One of the most significant changes to accessing will be based on a new metaphor for the web based on objects and classes, instead of on documents. Documents will still retain their importance, but will be viewed as one type of physical object rather than as the overriding metaphor for representing all forms of information.

[†]`http://www.yahoo.com`.

9. INTELLECTUAL PROPERTY AND ECONOMICS

If an author creates a digital engine object, then why should this author publish it? Isn't there a conflict with intellectual and industrial patent and copyright assumptions? And should the digital object be free or should a company charge for the object? We see the need for both commercial and public-domain digital objects. This would reflect the way that software is currently marketed over the web. A free version may be a "demo version" with the full version being sold for profit.

Could a digital engine maker disclose company secrets? This key issue is not only a concern to industry, but to all model builders. A model, and its enclosing object, reflects intellectual property similar to what is published in a technical paper. There are two concerns that we can address. The first deals with industrial objects. Industrial objects can be reverse engineered to see "what makes them tick." Therefore, releasing a certain amount of information in a published digital object may be reasonable given that it is information that can be easily obtained through other means. However, the reverse engineering of hardware may be expensive in terms of equipment and time, and so the manufacturer will want to ensure that a similar cost is levied against the purchaser for the digital equivalent.

It it possible to publish public information while securing private information, so that an automobile engine's overall performance can be modeled without exposing the internals of the dynamics and the parameter estimates. This can be done through "black box" approaches where the input-output behavior for a digital object is published, but the internal structure is obscured. Moreover, a company can make it impossible to see any model structure while allowing the model author to access the web site by providing input to its object. There is a wide range of possibilities from allowing others to copy the digital object and all of its internal models, to allowing users to copy only the highest level behavior, to allowing users to access only the behavior of the digital object without allowing any structural model access. Ultimately, the practice of a free marketplace will drive down the cost of digital objects and make them more accessible to everyone.

10. QUALITY CONTROL

What if the model author of the engine creates a digital engine that operates differently than the actual one? The automobile company could provide full access to an invalid model. We must have quality control measures in place to help us with this situation. The physical metaphor provides some help. Many consumer groups and institutions exist to protect consumers from bad products. Digital products will require similar groups and testing procedures. If a company knowingly markets a bad digital product, they will ultimately pay for this error in the marketplace. The digital object must be treated with the same level of quality control as the physical counterpart. In some cases, a company might make a mistake in production and a part or entire vehicle must be recalled. This type of recall is made easier with the digital product. It behooves the model authors to create valid, quality objects. It may be that anyone can publish a digital object but this is true of physical objects as well. The situation is somewhat more acute with a digital automobile since to create an automobile in the first place, one must have invested a fair amount of time and resources; however, a digital engine could be created by the neighbor down the street. One must learn to trust certain sources more than others based on past performance of prior digital objects. Also, we must have ways of verifying our sources, developers and producers with methods such as digital signatures, watermarks and encryption.

11. EXISTING STANDARDS

Standards are what make the post-industrial revolution work. Without standards for digital objects, we will be in the same situation as the rifle makers of the 19th century who made individual pieces without an assembly line or the benefits of reuse. There are no current standards for digital objects; however, there are movements in this direction.

The Unified Modeling Language (UML[3–5]) is a potential standard for representing object oriented software components, not necessarily physical systems. Due to its breadth in an attempt to address software in general, it is missing dynamic and geometric specifications for physical objects. The basic approach in UML with visual class structures is fruitful, however, and represents an excellent step in representing and visualizing class and object structure.

The High Level Architecture (HLA[6]), created by the Department of Defense, is a detailed specification supporting distributed execution of simulation code. Code may be legacy code or could have been generated from models. What is missing from HLA is any description of dynamic or geometric *models*. HLA's focus is on tying together a set of

heterogeneous components, each of which may represent a physical simulator, personal computer or a simulation with human participants. Therefore, reusability is at the code level for software components, without an attempt to provide a standard for create the software from models. Reuse of software is a tremendous help to those who are looking for large-scale objects to plug into their simulations, but software reuse does not help the model author to construct the dynamics of an object itself.

The VLSI Hardware Definition Language (VHDL[7]) is a structural and behavioral language for representing integrated circuits. There is room in a VHDL specification for structure and layout as well as for behavior. It is made specifically for electronic applications, and does not attempt a broader purpose. We can look at the benefits and issues raised within the VHDL community to determine how these might affect digital object reusability beyond the VLSI domain.

12. MOOSE[8] AND RELATED WORK

Our goal is to create a formal specification for digital objects with class hierarchies, objects, geometric and dynamic models. There is no limitation as to the domain of application. The Multimodeling Object Oriented Simulation Environment (MOOSE[‡]) is a software implementation that accepts a specification in the form of a Distributed Modeling Markup Language (DMML). We have created our own Conceptual Modeling Language (CML) and Dynamic Modeling Language (DML) based on the multimodeling methodology, and we plan to use the Virtual Reality Markup Language (VRML[9]) specification for the geometric models. Regarding CML, both the HLA Object Model Template and UML class definitions suggest starting points for the CML grammar. Related work in the representation of classes, objects and models has been done by several groups. For example, Hill[10] focuses on the role of objects for simulation. Also, Zeigler's group[§] at the University of Arizona[11,12] and Mattsson[¶] at the Lund Institute of Technology in Sweden[13,14] have published widely in the area of object-oriented structures for simulation and control.

13. SUMMARY

If simulation is to make significant steps on the web, and in cost reduction, we need to move toward a digital object view of knowledge. If we take our telescopes and try to look into the future of modeling, we might be put off by the complexity of what lies ahead for digital objects. The idea that scientists and engineers will forever want to create their own models and simulations, without the ability to plug-and-play with digital objects represents an unfortunate situation. The model author might feel that no existing web objects can possibly match his requirements, however, we have demonstrated concrete steps that can be taken to alleviate the problems of reuse. Reusing digital objects may be a more profitable enterprise than for software components since digital objects bear a one-to-one relation with their physical cousins, and these physical objects have been already demonstrated in the marketplace to have real value. We feel that we must take proactive steps in making digital objects and their web-based representations a reality if simulation is to progress as a field. Nothing will happen overnight, but we need to seek out the really hard problems and then address them one by one.

ACKNOWLEDGMENTS

I would first like to thank all of the students who make MOOSE a reality. These students have contributed very significant amounts of time toward the digital object methodology and its implementation. Robert Cubert, Gyooseok Kim Youngsup Kim, and Kangsun Lee are all Ph.D. candidates—and core members comprising the MOOSE team—in the CISE Department within the University of Florida. I would like to thank the following funding sources that have contributed towards our study of modeling and implementation of the MOOSE multimodeling simulation environment: GRCI Incorporated (Gregg Liming) and Rome Laboratory (Steve Farr) for web-based simulation and modeling, as well as Rome Laboratory (Al Sisti) for multimodeling and model abstraction. We also thank the Department of the Interior under a contract under the ATLSS Project (Don DeAngelis, University of Miami). Without their help and encouragement, our research would not be possible.

[‡] http://www.cise.ufl.edu/~fishwick/moose.html.

[§] Ref. http://www-ais.ece.arizona.edu/.

[¶] http://www.control.lth.se/~cace/.

REFERENCES

1. P. A. Fishwick, D. R. C. Hill, and R. Smith, *1998 International Conference on Web-Based Modeling and Simulation*, Society for Computer Simulation International, San Diego, CA, 1998. 203pp.
2. P. A. Fishwick, J. G. Sanderson, and W. F. Wolff, "A Multimodeling Basis for Across-Trophic-Level Ecosystem Modeling: The Florida Everglades Example," *SCS Transactions on Simulation* , 1997. To be published.
3. P.-A. Muller, *Instant UML*, Wrox Press, Ltd., Olton, Birmingham, England, 1997.
4. R. C. Lee, *UML and C++: A Practical Guide to Object-Oriented Development*, Prentice Hall, 1997.
5. C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*, Prentice Hall, 1998.
6. "DoD High Level Architecture (HLA)," 1998. `http://hla.dmso.mil/`.
7. J. Bhasker, *A VHDL Primer: Revised Edition*, Prentice Hall, 1995.
8. R. M. Cubert and P. A. Fishwick, "MOOSE: An Object-Oriented Multimodeling and Simulation Application Framework," *Simulation* , 1997. To be published.
9. B. Roehl, J. Couch, C. Reed-Ballreich, T. Rohaly, and G. Brown, *Late Night VRML 2.0 with Java*, Ziff-Davis Development Group, 1997.
10. D. R. C. Hill, *Object-Oriented Analysis and Simulation*, Addison-Wesley, 1996.
11. B. P. Zeigler, *Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, 1990.
12. B. P. Zeigler, *Objects and Systems: Principled Design with Implementations in C++ and Java*, Springer Verlag, 1997.
13. S. E. Mattsson and M. Andersson, "Omola—An Object-Oriented Modeling Language," in *Recent Advances in Computer-Aided Control Systems Engineering*, M. Jamshidi and C. J. Herget, eds., vol. 9 of *Studies in Automation and Control*, pp. 291–310, Elsevier Science Publishers, 1993.
14. S. E. Mattsson, "Towards a New Standard for Modelling and Simulation Tools," in *SIMS'93, Applied Simulation in Industry — Proceedings of the 35th SIMS Simulation Conference*, T. Iversen, ed., pp. 1–10, SIMS, Scandinavian Simulation Society, c/o SINTEF Automatic Control, (Trondheim, Norway), June 1993. Invited paper.