11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference
6 - 8 September 2006, Portsmouth, Virginia

AIAA 2006-7082

Object-Centered Engineering: A Methodology for Virtual Engineering

Douglas S. McCorkle* and Kenneth M. Bryden†
Iowa State University, Mechanical Engineering, Ames, IA, 50011

and

Daniel A. Ashlock‡
University of Guelph, Department of Mathematics and Statistics, Ontario, Canada, N1G2W1

[Abstract] In the product realization process, it is not currently possible to go from business case models to the final production system in the virtual design space. Virtual engineering aims to address this problem. Virtual engineering techniques will allow users to track the production of a product or system from birth to death, from the complete business case model to the customer's feedback on the first production run. To implement a framework that will handle the broad range of information that is necessary to track a product through its complete life cycle, an object-centered approach involving virtual objects is required. These virtual objects will represent the physical objects as they exist in the "real world." To enable these objects to have extensible qualities similar to object-oriented principles, similar techniques to those used by object-oriented design will be employed. Some of these techniques include multi-representational models, hierarchy, inheritance, and dynamic access. One important justification for an object-centered approach is that it gives stakeholders and engineers a mechanism for discourse regarding the product or system under design. Giving the engineer and other collaborators a comfortable and familiar mechanism by which to share and discuss ideas is crucial in allowing users to gain understanding about a product's key issues. These ideas and processes are embodied in virtual engineering and the method behind it, which is referred to as object-centered engineering. These ideas and the object-centered method will be discussed in this paper.

I. Introduction

Over the past decade, computing and software have significantly changed. The explosion of technology has redefined what engineers do and are responsible for in the product engineering process. Engineers are being required to understand and be capable of making informed decisions across a diverse set of topics. For this to occur, engineers must have the proper mechanisms and tools to access information about the system under design as well as an intuitive interface and virtual world in which to work and explore to enable well informed decisions. The virtual world also creates the mechanism by which other engineers and stakeholders can collaborate and have a common platform for discourse and, hopefully, for discovery. This is the idea behind developing object-oriented languages and paradigms for various topics. Objects give the engineer and user a mechanism by which to map the physical object in the real world to the virtual object in the virtual world. One important aspect of the virtual object as discussed in this document is that it is an actual duplicate of the object as it exists in the real world rather than a conceptual object or an object used in a computer program for data abstraction and manipulation. For example, the object has the same geometric representation and surrounding data as the physical object, and data that can be accessed for an object includes the mechanical information (Finite Element Analysis [FEA], dynamics, Computational Fluid Dynamics [CFD]), costing information and economics information. Once virtual objects are

* Research Assistant, Mechanical Engineering, 2274 Howe Hall, Rm 1620.
† Associate Professor and Associate Chair, Mechanical Engineering, 3030 Black Engineering.
‡ Professor, Department of Mathematics and Statistics, MACN Rm. 521.

created for physical objects, the engineer is able to access each object and all of its associated information and share that information with other objects and engineers in a platform that is common to all parties involved.

The method that will be described in this paper is expected to provide a foundation to accomplish virtual engineering. In the background section, the previous work related to object-oriented paradigms, computing, and engineering design will be described. In the objects section, the object-centered engineering method will be defined. In the results section, research progress will be described. In the conclusions and future work section, an overview of the method and an initial implementation will be proposed.

II. Background

The introduction alludes to the changes that engineers have adapted to with every major shift in technology and methodology. The present day is no different and requires that engineers continue to adapt paradigms and methodologies. Given that the environment in which engineers currently operate is rapidly changing and is constantly inundated with new technologies, it is necessary for a new paradigm in engineering to be developed. This paradigm is virtual engineering. Some of the new issues facing the engineering community are the advent of ubiquitous computing, software integration, and data avalanche. The ability of engineers to work in this environment is crucial to their ability to solve the problems of the current century, which are complex and have many unknown variables. To enable the implementation of a new paradigm to create this engineering framework, many tools need to be developed. While many of these tools are under development, many have yet to be investigated in detail. Some of the tools that are under development and being investigated include object-centered engineering, self-organization, inverse engineering techniques, optimization routines, game theory to control information flow, recomposable models, distributed collaborative engineering environments, object-oriented methodology (Ref. 4), (Ref. 5), (Ref. 9), (Ref. 16), (Ref. 17), (Ref. 28), (Ref. 32), (Ref. 33), (Ref. 34), and open interfaces for information transfer.

III. Objects

When software objects were first implemented by Nygaard and Dahl (Ref. 6), their purpose was to allow a more intuitive connection between the world and the program being created. This is the predominant theme of many discussions of objects (Ref. 13), (Ref. 14), (Ref. 19), (Ref. 20). Objects allow the individuals interacting with a system to more easily adapt to what the developer is trying to convey. Also, from a programming standpoint, objects easily allow the programmer to create a program with characteristics that more generally resemble the problem being simulated or solved, which is what Kay (Ref. 12), (Ref. 15), (Ref. 18), (Ref. 24) and Nygaard (Ref. 6) proposed.

When constructing a simulation, users should not be concerned with the solver or solvers that are employed, but should construct the part as in real life. This enables a narrative to be constructed (Ref. 8), (Ref. 27) starting with the conceptual economics study (Ref. 2), (Ref. 3), (Ref. 21), (Ref. 25), (Ref. 26), (Ref. 31) and ending with the detailed part in production. This process should be as seamless as reading a book, playing a game, or constructing any other digital document or digital information component. Accomplishing this will draw on many areas of current research as described previously. The main task of engineering objects is to create a way to manage complexity (Ref. 23) and to enable the dynamic creation and addition of information in the decision-making environment. As noted above, many different areas of research have utilized the object-oriented approach to help reduce complexity as well as to increase the ease of use of the tool being created.

The objects that will be used to manage complexity are different from programmatic objects in OOP and are based on objects as described by the French philosopher Michel Foucault. Foucault says that objects are “the extension of which all natural beings are constituted – an extension that may be affected by four variables. And by four variables only: the form of the elements, the quantity of those elements, the manner in which they are distributed in space in relation to each other, and the relative magnitude of each element” (Ref. 10). These objects carry with them context and meaning and the ability to be modified by the user. They differ from programmatic objects because they are defined to contain information packets that can be manipulated and reconfigured by the user at run-time, different representations of the virtual objects, and in most cases will be distributed across many compute resources. Most of all, an engineering object has the ability to self-discover and adapt to other objects that may need to exchange information with that particular instance of the object. Given these differences, virtual objects still build on the programmatic object-oriented principles in that virtual objects are modular, easily reused, extensible, polymorphic, able to support complex objects (i.e., objects can make up other objects), and can be loosely or tightly coupled to other objects.

These objects help manage complexity because they are, in most cases, tied to models that are built on experimental data or on traditional numerical models. This is a crucial element of virtual objects because to manage complexity, it is necessary to utilize models to enable users to ask questions and query information to develop an intuition and understanding of the information presented (Ref. 11). Once the user has a model, it is possible to simplify the model and information to a point where it is familiar to the user. At this point, the user can begin to access more of the complexity within the model. Once implemented, engineering objects as described here will allow the user to easily traverse from a simplified state of information to a complex state of information, which is necessary to build a true understanding of the information (Ref. 7).

To enable the connection of objects, tools must be utilized that allow objects to self-describe themselves to the world and to understand information presented to them. Such tools are being created for the semantic web, which enables computers and users to collaborate more easily. The semantic web's core technology is "the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming" (Ref. 22). The extensible markup language (XML) provides a format that allows data structures to self-describe and provides a means to represent the data that it contains in a format readable by humans. The ideas upon which the semantic web is founded, along with the technology that is used to implement it, provide a platform on which objects can be extended to create a web in which contextual information is readily accessible to engineers. They also provide a means by which the product development cycle can be completed in a manner unlike any before. In the end, the semantic web provides the core functionality on which engineering objects can be built and extended to enable engineers to become authors of products rather than highly trained technical robots.

When objects are constructed and connected into a network giving the end author the ability to interact with and explore various options for connectivity and interrelationships, the resulting network resembles the web. These particular networks are called scale-free networks (Ref. 1). A typical characteristic of these networks is that there are a few major hubs or master objects that have sub-objects and information sources feeding into the master objects. For example, if "link:vesuite.org" is typed into Google, 0 hits are returned, indicating that there are no webpages containing links to vesuite.org. If "link:amazon.com" is typed into Google, 1,240,000 hits are returned. This illustrates that Amazon is a hub of connections within the web and holds a much higher standing within the community, much like an object with many connections, because it is trusted and utilized by a broad range of web users. With an understanding of the resulting network created by multiple objects it becomes possible to characterize classes of engineering objects.

IV. Results

To illustrate some of the progress that has been made in implementing engineering objects, the XML schema used to represent and describe various aspects of an engineering object will be presented. The schema is composed of a few key XML element types. The first type is the XMLObject element:

```
<xs:complexType name="veXMLObject">
    <xs:attribute name="objectType" type="xs:string" use="optional" />
</xs:complexType>
```

This element type is the base for all other elements within the VE-Open (Ref. 29) schema, enabling any other element type within the schema to be imbedded or referenced in a generalized manner. It is more of a formality but allows the logic to be complete when embedding and referencing derived XMLObjects in other element types. The functionality that XMLObject permits will be illustrated below in veCommand.

The second element type is the veDataValuePair:

```
<xs:complexType name="veDataValuePair">
    <xs:attribute type="xs:ID" name="id" use="required"/>
    <xs:complexContent>
      <xs:extension base="veXMLObject">
        <xs:sequence>
          <xs:element name="dataName" type="xs:string" />
          <xs:choice maxOccurs="1" minOccurs="1">
            <xs:element name="dataValue" type="xs:anyType" />
            <xs:element name="genericObject" type="veXMLObject" />
          </xs:choice>
```

```
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

The veDataValuePair type holds a descriptive name about the data it contains as well as an XMLObject or raw xs:anyType. This flexibility allows veDataValuePair to be a generic container element that holds any form of data being processed by a particular object. Note that a veDataValuePair is a complete extension of an XMLObject. This extension permits a veDataValuPair to be embedded within another veDataValuePair.

The third data (element?) type is veCommand:

```
<xs:complexType name="vecommand">
    <xs:complexContent>
      <xs:extension base="veXMLObject">
        <xs:sequence>
          <xs:element name="command" type="xs:string" />
          <xs:element name="parameter" type="veDataValuePair" minOccurs="0"
                                                    maxOccurs="unbounded" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

This element type contains a descriptive name for the command in addition to an xs:sequence of veDataValuePairs. The structure of a command is constructed to allow any object to request or send a series of data with information about its use or potential application of the data contained within. This permits objects to be separate instances and allows computers to then decode the XML data structure and perform necessary actions as directed or received by a particular object. Since a veDataValuePair can contain any XMLObject that is derived for VE-Open, a veCommand can be used as the overall container to transmit information about objects and the attributes used to describe them.

V. Conclusion

The object-centered method aims to address many of the issues facing the current engineering design process by allowing the engineer to focus on engineering and not on information integration. To illustrate the proposed capability of the object-centered method, an initial implementation of the XML schema has been described. The schema is currently in active use by VE-Suite (Ref. 30) for the core communication and data transfer mechanism within VE-Suite. VE-Suite proposes to allow a broad range of problems to be addressed across many disciplines for the complete life of a product or system. This will allow engineers to focus on using the information provided by engineering models and other diverse information models to make decisions in the product realization process. The initial interface specification, VE-Open, will allow engineers to address these multi-disciplinary issues and to collaborate at a level that allows information to flow seamlessly from one design team to another. These features will allow virtual engineering to occur within a wide variety of engineering facilities. Through the implementation of the object-centered method, the problems experienced when collaborating within large design teams will hopefully become issues of the past. The object-centered method, when implemented across each step of the product realization process, will create environments where virtualized systems and parts can be analyzed and produced with far fewer costs being devoted to the design and development phase of the realization process.

To continue to expand VE-Open and adapt it to more applications and problems, many issues must be resolved. Among these issues is formalizing an initial ontology to describe basic information sources within the engineering process from a high level so that objects can properly adapt and connect to one another without manual coding or direction from an engineer. In addition, new extensions to the schema must be made to permit objects to contain and operate on a hierarchy of models to enable integration across information sources. To couple models of varying fidelities, information about numerical solving requirements must be published by the objects in addition to the fidelity and pedigree of the information provided by the object. As each of these issues is addressed, the ability to seamlessly construct an environment where an engineer can focus on authoring a product becomes a reality.

References

[1]Barabasi, Albert-Laszlo (2003). Linked: How Everything Is Connected to Everything Else and What it Means for Business, Science, and Everyday Life, New York: Penguin Group.

[2]Barbiroli, G. and A. Focacci (2003). "A techno-economic analysis of the results of product diversification in household appliance durables--evaluating concreteness." International Journal of Production Economics **83**(3): 257-278.

[3]Celik, A. N. (2002). "Optimisation and techno-economic analysis of autonomous photovaltaic-wind hybrid energy systems in comparison." Energy Conversion and Management **43**(18): 2453-2468.

[4]Cutkosky, M. R., R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber (1993). "PACT: An experiment in integrating concurrent engineering systems." Computer **26**(1): 28-37.

[5]Dabke, P., A. Cox, D. Johnson (1998). "NetBuilder: An environment for integrating tools and people." Computer-Aided Design **30**(6): 465-472.

[6]Dahl, O.-J. and K. Nygaard (1966). "SIMULA--an ALGOL-based simulation language." Communications of the ACM **9**(9): 671-678.

[7]Davis, J. (1999). "Improving intelligence analysis at CIA: Dick Heuer's contribution to intelligence analysis." Retrieved 8 December, 2005, from http://www.odci.gov/csibooks/19104/art3.html.

[8]Dörner, R., P. Grimm, D. F. Abawi (2002). "Synergies between interactive training simulations and digital storytelling: a component-based framework." Computers & Graphics **26**(1): 45-55.

[9]Duffy, A. H. B., A. Persidis, K. J. MacCallum (1996). "NODES: a numerical and object based modelling system for conceptual engineering design." Knowledge-Based Systems **9**(5): 183-206.

[10]Foucault, M. (1994). The Order of Things. New York: Vintage Books.

[11]Gharajedaghi, J., (1999). Systems Thinking: Managing Chaos and Complexity: A Platform for Designing Business Architecture. Burlington, MA: Butterworth Heinemann.

[12]Goldstein, I. P. and D. G. Bobrow (1980). "Extending object oriented programming in Smalltalk." ACM Conference on LISP and Functional Programming, Stanford University.

[13]Heim, J. A. (1997). "Integrating distributed simulation objects." 1997 Winter Simulation Conference, Atlanta, GA.

[14]Horváth, L. and I. J. Rudas (2004). "Some possibilities for integrated intelligent object based engineering modeling." 5th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest.

[15]Kay, A. C. (1993). "The early history of Smalltalk." The Second ACM SIGPLAN Conference on History of Programming Languages, Cambridge, MA.

[16]Männistö, T., H. Peltonen, A. Martio, R. Sulonen (1999). "Modelling generic product structures in STEP." Computer-Aided Design **30**(14): 1111-1118.

[17]Martino, T. D., B. Falcidieno, Stefan Haßinger (1998). "Design and engineering process integration through a multiple view intermediate modeller in a distributed object-oriented system environment." Computer-Aided Design **30**(6): 437-452.

[18]Metz, C. (2001). "The perfect architecture." PC Magazine September 2001: web article, http://www.findarticles.com/p/articles/mi_zdpcm/is_200109/ai_ziff10175 2001.

[19]Pidd, M. (1992). "Object orientation & three phase simulation." 1992 Winter Simulation Conference, Arlington, VA.

[20]Rothenberg, J. (1986). "Object-oriented simulation: Where do we go from here?" 1986 Winter Simulation Conference, Washington, DC.

[21]Schubak, G. E. and D. S. Scott (1995). "A techno-economic comparison of power systems for autonomous underwater vehicles." IEEE Journal of Oceanic Engineering **20**(1): 94-100.

[22]Semantic Web, http://www.w3.org/2001/sw/

[23]Sharpe, J. E. E. and R. H. Bracewell (2000). "Handling complexity in object based modeling and simulation." IEE Seminar on Tools for Simulation and Modeling, London.

[24]Shoch, J. F. (1979). "An overview of the programming language Smalltalk-72." ACM SIGPLAN Notices **14**(9): 64-73.

[25]Shohet, I. M. and M. Puterman (2004). "Flat roofing systems: towards integrated techno-economic analysis." Building Research & Information **32**(2): 165-173.

[26]Singh, D., E. Croiset, P. L. Douglas, M. A. Douglas (2003). "Techno-economic study of CO_2 capture from an existing coal-fired power plant: MEA scrubbing vs. O_2/CO_2 recycle combustion." Energy Conversion and Management **44**(19): 3073-3091.

[27]Skov, M. B. and J. Stage (2002). "Designing interactive narrative systems: is object-orientation useful?" Computers & Graphics **26**(1): 57-66.

[28]Toye, G., M. R. Cutkosky, L. J. Leifer (1994). "SHARE: A methodology and environment for collaborative product development." International Journal of Intelligent and Cooperative Information Systems **3**(2): 129-153.

[29]VE-Open, http://vesuite.org/doxygen.php

[30]VE-Suite, http://vesuite.org

[31]Vlachos, G. T. and J. K. Kaldellis (2004). "Application of gas-turbine exhaust gases for brackish water desalination: a techno-economic evaluation." Applied Thermal Engineering **24**(17): 2487-2500.

[32]Wong, A. and D. Sriram (1993). "SHARED: An information model for cooperative product development." Research in Engineering Design **5**(1): 21-39.

[33]Xue, D. and Y. Xu (2003). "Web-based distributed system and database modeling for concurrent design." Computer-Aided Design **35**(5): 433-452.

[34]Zha, X. F. (2000). "An object-oriented knowledge based Petri net approach to intelligent integration of design and assembly planning." Artificial Intelligence in Engineering **14**(1): 83-112.