2008

A Decade and More of UML: An Overview of UML Semantic and Structural Issues and UML Field Use

John Erickson

GUEST EDITORIAL PREFACE

A Decade and More of UML:
An Overview of UML Semantic and Structural Issues and UML Field Use

John Erickson, University of Nebraska - Omaha, USA

INTRODUCTION

More than 10 years ago in 1997, three modeling advocates brought together their own distinct techniques to forge UML (Unified Modeling Language), and the world of modeling was forever changed (Booch, Rumbaugh, & Jacobson, 1999, 2005). The Object Management Group (OMG) immediately adopted the new language as the standard for their newly expanded object-oriented (OO) modeling scope (OMG, 2008), and the stage seemed set for a modeling explosion with UML leading the way into a brave new world of more accurate and better performing systems.

The OMG quickly developed its model-driven architecture (MDA) and began to promote the idea that models could and should be independent of the platform that they were created to be used on, so that the model, rather than the platform, became the focus of the systems analysis and design process (OMG, 2008). In some ways, this new perspective could be seen as a precursor of the service architectures (see Chen, Zhou, & Zhang, 2006; Erickson & Siau, in press; Li, Huang, Yen, & Chang, 2007) that appear to be driving many systems development initiatives currently. At last, it seemed that there was some agreement on a number of the important issues revolving around the building of information systems, and real progress toward creating truly better systems seemed equally possible.

However, a number of UML researchers and users alike believe that Booch, Rumbaugh, and Jacobson's creation, rather than emerging from a careful mapping of their three distinct techniques onto a new and carefully designed and created metamodel, included a certain element of negotiation and concession, retaining or discarding at least some of the components of their old modeling techniques on a more personal or political basis. Whether that is actually the case or not is irrelevant to the issue of the semantic, diagrammatic, and notational inconsistencies in UML that many different researchers and, more importantly, practitioners have identified as difficulties in adopting and using UML.

This article highlights some of the important issues plaguing UML in terms of research and adoption, attempts to examine the current state of affairs regarding UML, and poses questions for the future of modeling in general and UML in particular. In the next section, the article will discuss research related to the semantic inconsistencies of UML. Since semantic inconsistencies can lead to comprehension difficulties, this issue is discussed in the subsequent section, which is then followed by a section examining the complexity of UML. Next, the article examines some past and current usage

patterns of UML and ends with conjectures on the future of modeling and UML.

Semantic Inconsistencies of UML

UML Version 1.X was composed of nine distinct diagrammatic techniques that could be broadly separated into three types: structural, behavioral, and interaction (which describes both the structure and behavior of the system). The intent was to create a series of diagrams that were related in terms of some information overlap, but distinct to focus on the specific issue that each diagramming technique was created to address. UML 2.0, formally adopted in 2005, expanded the number of diagramming techniques to 13, retaining the three broad classifications of structural, behavior, and interaction. Dobing and Parsons (2000) noted that there were semantic inconsistencies in using a behavioral technique (use cases) as a means of capturing process information and using that data to elicit the structure of OO constructs such as classes, sequences, and state machines. Opdahl and Henderson-Sellers (2002) examined the UML metamodel soon after and suggested that a number of changes involving abstract metaclasses and subclasses would improve the semantics of the language. While at least some of the suggestions were implemented in later versions of UML, it remains questionable as to whether some of the more basic semantic issues have been completely resolved.

One of the criticisms of UML is that it is not semantically articulate enough to present unambiguous information to its users. In other words, many UML symbols can be interpreted in a variety of ways, leaving much subjectivity in interpretation. Jon Whittle (2000) at the NASA Ames Research Center proposed that the semantics of UML be standardized and that "formal methods be used to analyze UML models." Whittle further proposed that the reason for what he saw as the slow progress toward formalizing UML was that the semantics of UML were inherently informal.

Further evidence that the semantics of UML were inconsistent or informal, and indeed constituted a major problem or stumbling block when learning the language, came from Siau and Tian (2001, 2002), Shen and Siau (2003), and Siau and Loo (2006). Lange (2006) made a call for modeling standards or norms for UML in order to minimize the inconsistencies that often arise when one person or group creates a diagram and different people or groups try to use it. If this type of problem is still occurring after the release of UML 2.0, it likely means one of two things: either people are still using UML 1.X and encountering the same problems as in the past, or they have converted to UML 2.0 but some semantic inconsistencies still remain in the language. Whichever is the case, one means developers could use to address the issue is to try to agree up front on diagramming and notation conventions. While not eliminating the semantic inconsistencies of the modeling language, UML or other, it will at least minimize the problem for each specific instance or case. In the long term, it would of course be better if the industry could reach some agreement on standards, or better still, if the language proponents could develop such standards themselves.

Comprehension Issues Regarding UML

Dobing and Parsons (2000) noted that different facets of use cases could result in different models. For example, they found that use cases differed by content, format, comprehensiveness, detail, and methodology. Add to that the semantic issues, and people from a variety of perspectives (from developer to user, for example) can be severely disadvantaged when they try to understand and employ use cases. Dobing and Parsons ended with the conclusion that use cases may be inadequate for eliciting classes, and called for more research on the topic.

Similarly, Kim, Hahn, and Hahn (2000) studied diagrams and cognitive diagrammatic reasoning and enforced interpretation time limits on the participants in their experiment. They found that if multiple diagrams were used to convey information to users, a more thorough comprehension of the system resulted. In particular, visual cues and context were important considerations for thorough comprehension, and referring back once again to the semantic inconsistencies, if the language

symbols are inconsistent, this means that the analysis (and subsequent design) will more likely be inaccurate.

Siau and Lee (2004) found that use cases were necessary components of UML. They measured comprehension for both class and use-case tasks in an experiment and found that, for the same system, people given both class diagrams and use-case diagrams obtained additional information over people who were given only one diagram or the other. However, one of the problems the experiment encountered was that different people interpreted the same diagrams differently, with a wide variety of results.

Gemino and Wand (2003) took a higher level approach to examining the comprehension of models. They developed an evaluation framework that was independent of technique and concluded with recommendations that (a) theoretical and empirical approaches should be used to evaluate modeling techniques, (b) simply putting information on or into a model does not guarantee comprehension or understanding, and (c) domain expertise of the modelers is an important element of comprehension, so the evaluations should be comprised of problem-solving tasks that the modelers are familiar with. In other words, learning the domain and learning about the modeling language simultaneously may exacerbate comprehension problems.

Finally, theories and experiments originating in cognitive psychology are finding their way into use by MIS (management information system) researchers (e.g., Siau & Tan, 2005, 2006a, 2006b, 2006c; Siau & Wang 2007). Yusuf, Kagdi, and Maletic (2007) used eye-motion tracking equipment to learn how participants used the information provided in class diagrams to comprehend the diagram, noting that other contextual information (domain) and semantic information (class stereotypes) enhanced the comprehension process. Research such as this can help analysts create better diagrams that are easier for developers and clients to understand. As with the semantic and notational inconsistencies, if the language proponents and industry could put more work into the constructs of UML, perhaps some of the comprehension issues would be reduced.

MODELING-LANGUAGE COMPLEXITY ISSUES

There is probably little doubt, even among the most expert UML analysts and experts, that the modeling language is indeed complex. Worse, UML's complexity along with its semantic inconsistencies can easily and negatively affect developers', users', and others' comprehension of not only the diagrams and models, but also the systems themselves, resulting often in substandard systems.

Some of the first work on measuring modeling-method complexity was that of Rossi and Brinkkemper (1996). They developed a set of metrical instruments that were created to be independent of modeling technique or language, and measured the structural complexity of each diagramming technique. Siau and Cao (2001) used the metric set to analyze UML and 36 other diagramming techniques from 14 modeling languages (or techniques), both in aggregate and individually. They found that UML was 2 to 11 times more complex than any of the other modeling languages.

More work on the issue of complexity was done by Siau, Erickson, and Lee (2002, 2005). They attempted to separate complexity into subcomponents, which they called practical complexity and theoretical complexity. According to their research, theoretical complexity should be assessed by including all possible (metamodel) constructs in the metrical analysis, while practical complexity should be seen as comprising only those constructs that users actually were familiar with and regularly used. The research findings did indicate that using all possible constructs as a simple measure of complexity was not really an adequate explanatory vehicle for the complexity that UML users commonly face. Another result was that the most used constructs formed a core or kernel of UML; this will be an important issue discussed later.

Further investigation along this line of research involved examining the use of UML constructs in specific domain areas. Erickson and Siau (2004, 2007a, 2007b) studied the use of UML in real-time enterprise and Web-based systems. The general findings indicated

that there was a relatively stable core of UML that the research participants agreed upon via a Delphi study. This stability provided some support for the idea that a small core of UML could be used to assess the complexity of the language and also to suggest means of educating users as well as actually using the language in practice.

The research indicated that UML 1.X was structurally quite complex, and UML 2.0 introduced four new diagramming techniques, so there seems to be little doubt that additional structural complexity is a companion of the new diagrams individually and the language as a whole. At this point, little can be directly done to ameliorate the complexity built into the language itself, so short-term efforts should focus on means to deal with the complexity from a behavioral perspective.

Use of UML in the Field

Many case studies and experience reports on the use of UML in various domains and in many industries dating from early in its life until the present have been reported in a huge number of outlets. For example, Field, Heim, and Sinha (2004) created a UML-based process model for management and assessment of electronic service quality. Specifically, they used use-case diagrams to develop the process model and as such their effort used only one of the possible UML diagramming techniques. Mammar and Laleau (2006) used UML to develop UB2SQL, which is a tool for designing and developing database applications. Another example involved a large online securities trading company. The company indicated that it used UML in its IT development processes (Erickson, 2008). Through interviews with the company's IT employees actually doing the work, it was found that they were using UML to simply document the as-built systems; the company had created a systems development plan that the employees used each time they built a system, and that plan was not necessarily UML based. While the research there is on going, the company declines to be directly identified because of what they consider to be the proprietary nature of their systems structures and development processes. Nevertheless, the company insists that it makes only documentary use of UML component or deployment diagrams.

The OMG, while no doubt grinding its own ax, lists a number of companies or organizations that have been successful at a UML-based systems development process (OMG, 2008). They include ARINC Inc., Armstrong Consulting, EMC, the Trane Company, Xerox, a unit of Sony, and Charles Schwab, among many others. Information is not provided on how UML was used in those cases, although since they appear as success stories, the assumption must be that they make more comprehensive use of UML than some of the other more casual-use companies or organizations.

Research on how UML is actually used in the field is a bit more limited than the success stories and case studies previously mentioned. Erickson and Siau (2004, 2007a) conducted a Delphi study to try to determine how people actually use UML in the field. They found that the four most commonly used UML diagrams were, in order of importance and disregarding specific domains, class, use case, sequence, and state chart. If domain was included, in this case real time, Web based, and enterprise, three of the four diagrams remained the same in terms of perceived user importance: class, use case, and sequence. Dobing and Parsons (2008) conducted an industry study surveying system developers regarding their UML usage and found that the overall patterns were quite similar to those found by Erickson and Siau (2007a). Dobing and Parsons' 2008 study was endorsed by the OMG, making it a quite robust effort in terms of the number and variety of respondents.

Combine these two studies' results with the general usage described previously (Erickson, 2008; Field et al., 2004), and a relatively consistent theme emerges. Unless a company or organization is committed to fully using all the components and features of UML they are capable of using, they will likely in practical situations use a relatively small and usually fairly well-defined subset of the UML diagramming components, consisting of use-case and class diagrams, and sequence and state-chart diagrams as domain and circumstances dictate.

A second theme is more conjecture at this point, but could be included relatively easily in future research agendas. Companies more committed to fully using a majority of the components of UML will tend to be larger and have more experience in larger development projects. A reason for this might be that to create a fully expressive UML model, it is necessary to use the IBM/Rational Rose development tool. At more than $8,000 per user license, the cost of a full UML model in a model-driven architecture would limit the corporate and organizational users to those who can afford such development tool cost. Of course, other less expensive modeling tools exist, such as Microsoft Visio, but many, if not most of them, cannot be used to create models expressive enough for use in an MDA project.

POSSIBILITIES FOR FUTURE OF MODELING AND UML

Some will no doubt insist that in a world of Web services and service-oriented architectures, soon we will no longer have any need to model systems. The pieces of future systems will all exist independently in component libraries and repositories, and when companies compose their new systems, they will simply select and arrange the necessary modules or components and bind them to their own needs. Some predicted a similar fate for programmers when the OO paradigm was first embraced by the software engineering community. Once modular software code was written and deposited into code libraries, programmers would no longer be necessary.

As it has developed, programmers appear to be in as much demand now as they were before the OO explosion; they have simply shifted into Web development and other areas not thought of previously. Similarly, it seems quite likely that the need for systems analysis and design, and by necessity of analysis and design, modeling will remain quite high for the foreseeable future. Service-oriented architectures and Web services cannot compose, arrange, and deploy themselves, nor can they react to changing conditions or plan proactively. In the same way, not every company or organization configures an enterprise resource planning system exactly the same way—even close competitors in the same industry.

While the need for modeling in the future seems to be a fairly safe bet, UML and all other modeling languages for that matter should not assume that they are in the same situation. Certainly there will be need for modeling in the future, but will there be a need for UML or other specific modeling techniques? The modeling languages and techniques must adapt to the needs of their users as spoken human languages change in response to the needs of their speakers. It has been said that a language that does not change is a dead language.

Change for the sake of change itself is not usually considered to be a wise course of action, so directed change might be an appropriate venue. The primary problems regarding UML appear to manifest in four areas, at least in this exposition. First, some effort should be expended toward alleviating, to the extent possible, the inconsistent semantics and notational vagueness that have plagued users for the last 10 years. This is not meant to downplay the efforts that have been made in this area, but to suggest that continuing work is needed.

Second, comprehension of UML diagrams has been a problem for most of the 10 (or 11) years of UML's existence. Comprehension is related to the semantic and notational problems, but also to the UML users' experience, as well as the complexity inherent in the language. User experience is important not only in domains, but also in terms of UML experience. Neither are problems that UML itself can be changed to address. Rather, the issue is one of education and marketing. UML needs to continue to sell itself to its users and to motivate them to learn the language or risk being outsold by competing modeling techniques.

Third, UML is extremely complex, and in the short term, is not likely to become less complex. This means that user training and experience with UML is doubly important because it addresses two areas of concern. However, some thought should be given in the long term to finding ways to allow UML to appear less complex, if not actually be less complex. While this may be a pipe dream, the perceived

complexity of UML might very well represent a major stumbling block to adoption of UML by developers and others intimately involved in the systems development process.

Finally, UML is used by many developers and organizations as a way to document the systems analysis and design of various hardware and software components. The models are typically used for informational and documentation purposes and are not developed to the point where they are complete or expressive enough to comprise a model-driven architecture. While this is not necessarily a problem inherent in the UML metamodel, semantics, or notation, and may not even be a problem at all, the OMG and other interested parties should be aware that UML is probably not used fully by the majority of its users. In spite of this, UML has become a much used and useful tool for many organizations and people, and in some ways is now the lingua franca of modeling.

REFERENCES

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The unified modeling language user guide.* MA: Addison-Wesley.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The unified modeling language user guide* (2nd ed.). MA: Addison-Wesley.

Chen, Y., Zhou, L., & Zhang, D. (2006). Ontology-supported Web service composition: An approach to service-oriented knowledge management in corporate services. *Journal of Database Management, 17*(1), 67-84.

Dobing, B., & Parsons, J. (2000). Understanding the role of use cases in UML: A review and research agenda. *Journal of Database Management, 11*(4), 28-36.

Dobing, B., & Parsons, J. (2008). Dimensions of UML diagram use: A survey of practitioners. *Journal of Database Management, 19*(1), 1-18.

Erickson, J. (2008). *Use of UML at an on-line securities trading company* [Working paper]. Omaha, NE: University of Nebraska-Omaha.

Erickson, J., & Siau, K. (2004). *Toward practical measures of complexity in real-time modeling methods.* Paper presented at the Americas Conference on Information Systems.

Erickson, J., & Siau, K. (2007a). *Can UML be simplified? Practitioner use of UML in separate domains.* Paper presented at Exploring Modeling Methods in Systems Analysis and Design (EMMSAD) 2007 Workshop, Trondheim, Norway.

Erickson, J., & Siau, K. (2007b). Theoretical and practical complexity of modeling methods. *Communications of the ACM, 50*(8), 46-51.

Erickson, J., & Siau, K. (2008). Web services, service oriented computing, and service oriented architecture: Separating hype from reality. *Journal of Database Management, 19*(3), 42-54.

Field, J., Heim, G., & Sinha, K. (2004). Managing quality in the e-service system: Development and application of a process model. *Production and Operations Management, 13*(4), 291-306.

Gemino, A., & Wand, Y. (2003). Evaluating modeling techniques based on models of learning. *Communications of the ACM, 46*(10), 79-84.

Kim, J., Hahn, J., & Hahn, H. (2000). How do we understand a system with (so) many diagrams? Cognitive integration processes in diagrammatic reasoning. *Information Systems Research, 11*(3), 284-303.

Lange, C. (2006). Improving the quality of UML models in practice. *International Conference on Software Engineering*, pp. 993-996.

Li, S., Huang, S., Yen, D. C., & Chang, C. (2007). Migrating legacy information systems to Web services architecture. *Journal of Database Management, 18*(4), 1-25.

Mammar, A., & Laleau, R. (2006). UB2SQL: A tool for building database applications using UML and B formal method. *Journal of Database Management, 17*(4), 70-89.

Object Management Group (OMG). (2008). Retrieved from http://www.omg.org/mda/faq_mda.htm, on 1-14-2008

Opdahl, A., & Henderson-Sellers, B. (2002). Ontological evaluation of the UML using the Bunge-Wand-Weber model. *Software and Systems Modeling, 1*(1), 43-67.

Rossi, M., & Brinkkemper, S. (1996). Complexity metrics for systems development methods and techniques. *Information Systems, 21*(2), 209-227.

Shen, Z., & Siau, K. (2003). An empirical evaluation of UML notational elements using concept

mapping approach. *Proceedings of the International Conference on Information Systems (ICIS)*, Seattle, WA (pp. 194-206).

Siau, K., & Cao, Q. (2001). Unified modeling language (UML): A complexity analysis. *Journal of Database Management, 12*(1), 26-34.

Siau, K., Erickson, J., & Lee, L. (2002). Complexity of UML: Theoretical versus practical complexity. *Workshop on Information Technology and Systems (WITS)*, Barcelona, Spain (pp. 7-12).

Siau, K., Erickson, J., & Lee, L. (2005). Theoretical vs practical complexity: The case of UML. *Journal of Database Management, 16*(3), 40-57.

Siau, K., & Lee, L. (2004). Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML. *Requirements Engineering, 9*(4), 229-237.

Siau, K., & Loo, P. (2006). Identifying difficulties in learning UML. *Information Systems Management, 23*(3), 43-51.

Siau, K., & Tan, X. (2005). Improving the quality of conceptual modeling using cognitive mapping techniques. *Data and Knowledge Engineering, 55*(3), 343-365.

Siau, K., & Tan, X. (2006a). Cognitive mapping techniques for user-database interaction. *IEEE Transactions on Professional Communications, 49*(2), 96-108.

Siau, K., & Tan, X. (2006b). Use of cognitive mapping techniques in information systems. *Journal of Database Management, 17*(3), i-iv.

Siau, K., & Tan, X. (2006c). Using cognitive mapping techniques to supplement UML and UP in information requirements determination. *Journal of Computer Information Systems, 46*(5), 59-66.

Siau, K., & Tian, Y. (2001, December 16-19). The complexity of unified modeling language: A GOMS analysis. In *Fourteenth International Conference on Information Systems (ICIS 01)*, New Orleans, LA (pp. 443-448).

Siau, K., & Tian, Y. (2002, December 14-15). Analyzing unified modeling language using GOMS. In *Twelfth Workshop on Information Technology and Systems (WITS 02)*, Barcelona, Spain (pp. 7-12).

Siau, K., & Wang, Y. (2007). Cognitive evaluation of information modeling methods. *Information and Software Technology, 49*(5), 455-474.

Whittle, J. (2000). Formal approaches to systems analysis using UML: An overview. *Journal of Database Management, 11*(4), 4-13.

Yusuf, Kagdi, & Maletic. (2007). Assessing the comprehension of UML class diagrams via eye tracking. In *Fifteenth IEEE International Conference on Program Comprehension (ICPC '07)* (pp. 113-122).

John Erickson is an assistant professor in the College of Business Administration at the University of Nebraska at Omaha. His research interests include UML, software complexity and Systems Analysis and design issues. He has published in journals such as the CACM, JDM, and in conferences such as AMICIS, ICIS WITS, EMMSAD, and CAiSE. He has also co-authored several book chapters.