December 2004

Theoretical and Practical Complexity of UML

John Erickson
University of Nebraska at Omaha

Keng Siau
University of Nebraska-Lincoln

Theoretical and Practical Complexity of UML

John Erickson
University of Nebraska at Omaha
johnerickson@mail.unomaha.edu

Keng Siau
University of Nebraska-Lincoln
ksiau@unlnotes.unl.edu

ABSTRACT

Systems development methods have become more complex, concurrently with today's systems. UML (Unified Modeling Language) has been criticized for its complexity, for those learning and using it. Using Rossi and Brinkkemper's (1996) metrics, Siau and Cao, (2001) completed a complexity analysis of UML and other modeling techniques, finding that UML is more complex than other techniques. Siau, Erickson and Lee (2002) argued that Rossi and Brinkkemper's metrics present the theoretical maximum, as opposed to a practical complexity, which must be less than the maximum. Therefore, Siau and Cao's UML complexity analysis represents the theoretical complexity of UML. The current research proposes that a subset of UML (a kernel) composed of the most commonly used constructs, would more closely represent the complexity that practitioners face when using the language. A Delphi study is conducted using practitioners as experts, in an attempt to identify a use-based UML kernel and UML's practical complexity.

Keywords

UML, complexity, complexity metrics, Delphi study, modeling method metrics.

INTRODUCTION

Our world becomes more complex every day. Technology and system developers promise a reduction in life's everyday burdens because of technological advances. While perhaps some of that has proved real, systems have expended to encompass more and more of the tasks once performed manually. It should come as no great surprise that systems are now more complex than ever. For example, the latest versions of Microsoft Windows now approach 100 million lines of code; that alone should alert us to the realities of software complexity.

If systems and software have become more complex, then is not out of line to suppose that the underlying systems development process has become more complex concurrently. Further, it should be reasonable to presume that specific development methods have paralleled our seemingly inexorable march toward greater and greater complexity. An example is the soon-to-be-released UML (Unified Modeling Language) 2.0. While the original UML 1.X has been criticized for its complexity, inconsistent semantics, and ambiguity (Dobing and Parsons, 2000; 2002; Zendler, Pfeiffer, Eicks, and Lehner, 2001), the early versions also lacked truly useful extension mechanisms that would facilitate use in a variety of settings. UML 2.0, purportedly addresses the semantics, ambiguity and extension issues, but likely at the expense of additional and continued complexity.

UML is an Object-Oriented modeling language that aids developers by allowing creation of models representing the system being developed. Siau, Erickson and Lee (2002) proposed that Rossi and Brinkkemper's (1996) metrics present the theoretical maximum complexity, as opposed to a practitioner-based complexity, that must be less than the theoretical maximum. Theoretical complexity, per Rossi and Brinkkemper metrics, includes all of the possible constructs that can be used to create models. Since all the constructs are included, the resulting metrics represent the theoretical complexity. The current research proposes that a subset of UML, a kernel or practical type of complexity, composed of the most commonly used constructs, would be more representative of the complexity that practitioners face when using the modeling language. We refer to this as practical complexity.

In this research, a practitioner-based kernel of UML is developed via a Delphi study, and initial results are presented herein. The application of the results using a metrical analysis technique is in process at the time of this writing but we will available for presentation in August. This paper represents a research in process.

LITERATURE

Rossi and Brinkkemper's (1996) research proposed and developed a relatively easy to use and straightforward means to quantitatively measure system development methods. Specifically, the metrics are based on metamodel techniques, and purport to measure the complexity of the method under analysis. According to Rossi and Brinkkemper (1996), complexity is critical to measure because researchers believe complexity to be closely related to how easy a specific method is to use and to learn.

Siau and Cao's (2001) research applied Rossi and Brinkkemper's complexity metrics to UML. Their reasoning for using the particular metric set is that they contend that the metrics are among the most comprehensive, and that Rossi and Brinkkemper's approach "…have been used to evaluate the ease-of-use of OO techniques." Siau and Cao (2001) also compared UML's complexity with 36 OO techniques from 14 methods, as well as each of the 14 methods in aggregate.

One of Siau and Cao's (2001) noteworthy findings is that UML is far more complex (from 2 to 11 times more complex) in aggregate than any of the other methods. The relative overall complexity highlights one of the issues regarding UML, with the result that it can appear overwhelming to those new to UML (See Figure 1). Additionally, when human cognitive limitations to short term memory are added to this mix, UML can appear even more difficult to master.

Diagram	**N (O_T)**	**n (R_T)**	**N (P_T)**	**$\overline{P}_O$ (M_T)**	**$\overline{P}_R$ (M_T)**	**$\overline{R}_O$(M_T)**	**$\overline{C}$ (M_T)**	**C\`(M_T)**
Class	7	18	18	1.71	1.22	2.57	0.1	26.40
Use Case	6	6	6	1	0.83	1	0.17	10.39
Activity	8	5	6	0.75	0.2	0.63	0.13	11.18
Sequence	6	1	5	0.67	6	0.17	0.13	7.87
Collaboration	4	1	7	1	8	0.25	0.14	8.12
Object	3	1	5	1.67	3	0.33	0.33	5.92
StateChart	10	4	11	1	0.5	0.40	0.09	15.39
Component	8	10	9	1	3.6	1.25	0.11	15.65
Deployment	5	7	5	1	1.14	1.40	0.2	9.95

Figure 1

Legend:
n (O_T) – count of object types per technique.
n (R_T) – count of relationship types per technique.
n (P_T) – count of property types per technique.
$\overline{P}_O$ (M_T) – average number of properties for a given object type.
$\overline{P}_R$ (M_T) – average number of properties per relationship type.
$\overline{R}_O$(M_T) – number of relationship types that can be connected to a certain object type.
$\overline{C}$ (M_T) – average complexity for the entire technique.
C\`(M_T) – total conceptual complexity of the technique.

THEORETICAL FOUNDATION

Humans generally have cognitive problems processing information that is overly complex (Anderson and Lebiere, 1998; Miller, 1956). This problem surfaces often as people build information systems, which tend to be extremely complex. Anderson and Lebiere's Atomic Components of Thought (ACT) breaks knowledge into declarative knowledge and procedural knowledge. Declarative knowledge is similar to that knowledge captured in a dictionary – it is a list of what we know. Procedural knowledge, on the other hand is knowledge about how things work. Procedural knowledge depends on declarative knowledge as a starting point, but uses that knowledge to help solve problems (Anderson and Lebiere, 1998). Declarative knowledge is produced in chunks, and is constrained by our cognitive limits (Miller, 1956). Procedural knowledge is used to create production rules that describe productions, or specific steps we use to solve common and complex problems (Anderson and Lebiere, 1998). See Figure 2.

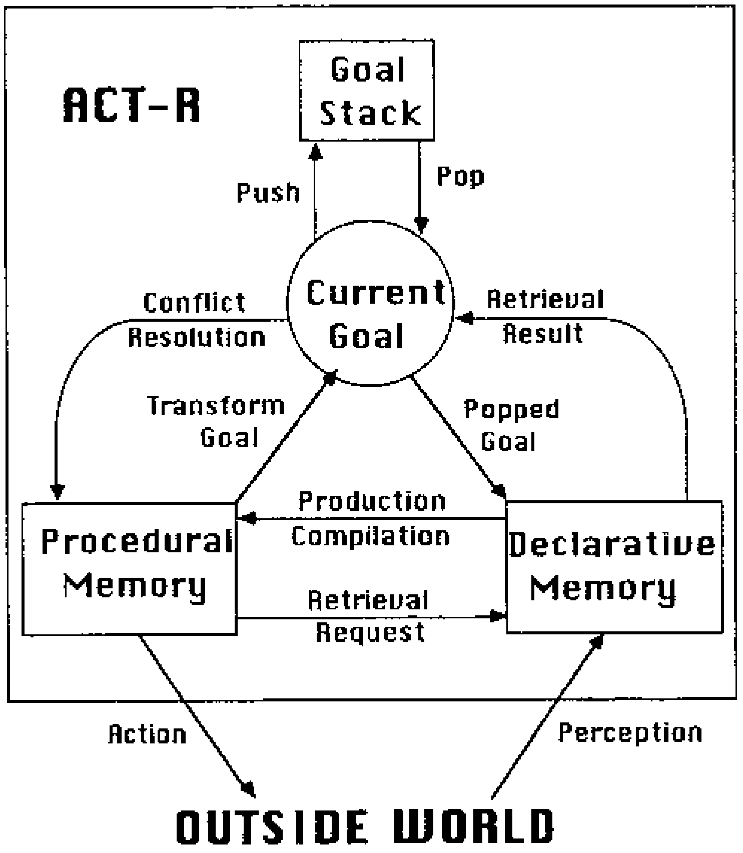

Figure 2 (Source Anderson and Lebiere, 1998)

Complexity can take many forms. For the purposes of this research, complexity will be approached from two separate but closely related perspectives; cognitive complexity as related to human perception, and structural complexity, as related to the structural properties of the diagramming techniques found in modeling approaches such as UML diagrams. In this context cognitive complexity can be defined as the mental burden people face as they work with systems development constructs.

The research proposes to adopt the ideas on regarding structural complexity proposed by Briand, Wüst, and Lounis (1999), in which the physical (structural) complexity of diagrams affects the cognitive complexity faced by the humans using the diagrams as aids to understand and/or develop systems. (See Figure 3)

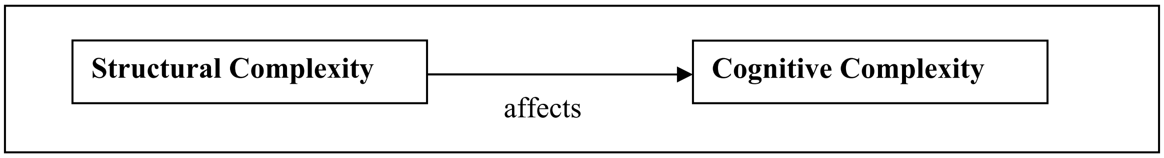

Figure 3 (Adapted from Briand, Wüst, and Lounis, 1999)

Since cognitive complexity is difficult and arguably even impossible to measure, structural complexity will be used to explain cognitive complexity. Structural complexity can be defined as a function of the number of distinct elements (or constructs) that constitute a given diagramming technique. Rossi and Brinkkemper (1996) formulated seventeen distinct definitions relating to the structural complexity of each diagramming technique. Using all available constructs (the metamodel component); these definitions form an estimate of the total structural complexity of the diagramming technique, which this research terms theoretical complexity.

Structural complexity is a part of the structural characteristics of the information or modeling system, and for this research refers to the elements, or constructs that comprise a given diagramming technique. These constructs would include meta-construct types such as objects (classes, and interfaces), properties (class names, attributes, methods, and roles), and relationships and associations (aggregations generalizations, specializations).

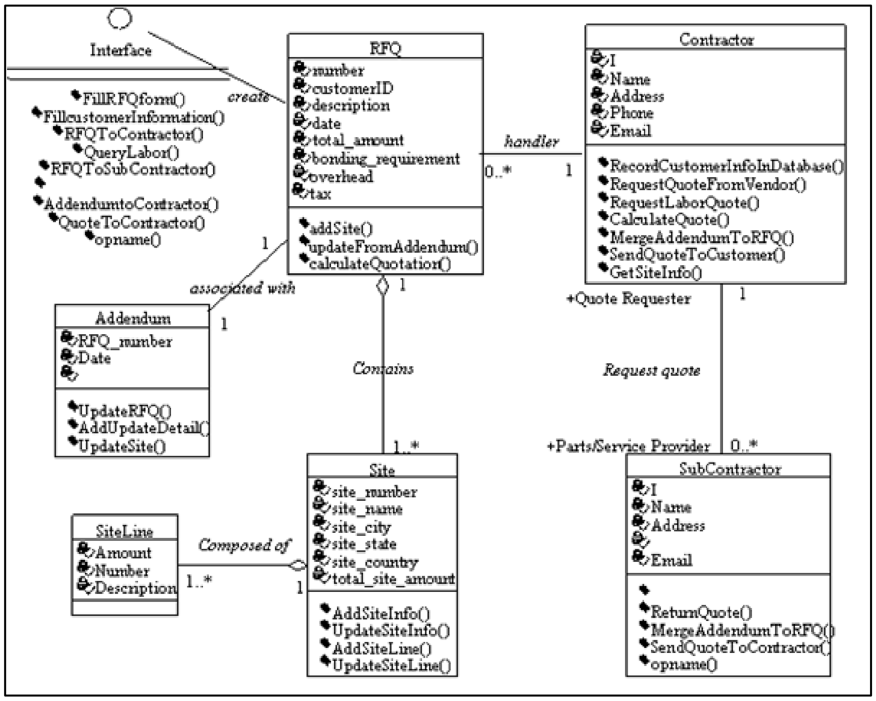

Figure 4 (Erickson and Siau, 2003)

For example, the class diagram in Figure 4 uses a number of constructs, in presenting the information it represents. There are six classes, and one interface. The classes each have a name, attributes, and methods. In addition, the class has relationships with other classes, and also with the Interface. However, the diagram does not make use of all the constructs that could be used in a class diagram. The Rossi and Brinkkemper metrics serve as the operational definition (as well as a measure) of the structural complexity of diagrams, which we propose represents theoretical complexity. The current research proposes to define practical complexity as a subset of the Rossi and Brinkkemper metrics, based on the practitioner identification, and artifact-based validation of the UML kernel.

RESEARCH QUESTION

Siau *et al*. (2002) provided evidence indicating that the theoretical metrics (total structural complexity) might not be representative of the complexity practitioners' face when using UML class and use case diagrams. This result motivates the research question:

Can an alternative set of complexity measures based on practitioner usage patterns be defined?

In other words, instead of including all the constructs available in a modeling method in computing the method complexity, the proposed research will base the complexity computation based on the usage patterns of the modeling constructs. The research objectives for this study are, (i) determine the most commonly used constructs in the UML diagrams and, (ii) use the results to define or identify an alternative to the established metrics (i.e., theoretical complexity metrics), and propose that they represent a kernel of UML (i.e., practical complexity metrics).

RESEARCH METHODOLOGY

The research investigates practical complexity, and the formulation of a UML kernel by means of a Delphi study. Delphi studies attempt to form a reliable consensus of a group of experts in a specialized area (Ludwig, 1997). The approach is a process that focuses on collecting information from the expert group though a series of questionnaires, and providing feedback to the group between questionnaires. Because the research is aimed at identifying the complexity that people face when using UML, a Delphi study composed of a group of practitioners with at least two years of UML experience form a better platform than other approaches (e.g., survey or case study) for identifying a UML kernel. Usually the group of experts is geographically dispersed, as is the case for many of the subjects participating in this research (i.e., the different organizations and people involved). The questionnaires are designed to allow the collection of expert opinions on the subject, and then to facilitate the refinement or focus of the subsequent versions to narrow in on a consensus.

PRELIMINARY RESULTS

Based on respondent ratings, the final diagram importance ratings are presented in Table 2 below. The experts rated relative diagram importance on a 5-point Likert type scale with 1 being the most important and 5 the least important. The top 4 diagrams in terms of mean rating scores were also selected by the respondents to be included in the kernel. The next highest rated diagram in terms of importance attained only a 31% consensus level, indicating the respondents clearly distinguished between important and less important diagrams, and kernel and non-kernel diagrams. Results are available for the nine diagrams individually, but are not presented here because of space limitations.

The results (see Table 1) indicate that practitioners clearly distinguish between what they consider to be the more important and less important UML diagrams (and diagram constructs). This can be thought of as a measure of practical complexity, which practitioners could use as a starting point to train UML newcomers. From a research perspective, the research results could be used to further the development of practical complexity measures, and to verify the results by examining implementation artifacts in various organizations and across system types. The results could also be helpful in the future development of UML.

Construct	Mean	Standard Deviation	% "Yes" for Kernel
Class	1.00	0.00	100.0%
Use Case	1.61	0.79	90.9%
Sequence	1.73	0.70	95.5%
Statechart	1.81	0.51	100.0%
Component	2.31	0.70	31.8%
Activity	2.41	0.55	27.3%
Collaboration	2.57	0.87	22.7%
Deployment	2.69	0.75	9.1%
Object	3.00	0.86	9.1%

Table 1 UML Diagram results (Shading indicates kernel)

DISCUSSION

If we include just the first 4 diagrams in the kernel, and assume that it represents the most commonly used diagrams in UML, then the structural complexity (Briand, Wüst, and Lounis 1999) and related cognitive complexity that users experience should be more manageable than if the users were commonly using all of the diagrams, and all of their related constructs to model systems. If all of the constructs associated with the non-kernel items also "disappear", that should represent a significant (non-statistical) reduction in complexity. This research identifies 116 total constructs in the nine primary UML diagrams, and 58 of those are included in the top 4 importance rated diagrams. Of those 58 constructs, 31 were identified by respondents as kernel items, and would make up the UML kernel. This results in 26.7% of the total number of UML constructs comprising the respondent identified UML kernel.

While a smaller number of constructs may be an indicator of less complexity, using the Rossi and Brinkkemper metrics should provide a more concrete analysis of the kernel. This analysis is currently in progress and will be available by the conference date.

REFERENCES

1. Anderson, J., and Lebiere, C. (1998) *The Atomic Components of Thought*, Lawrence Erlbaum Associates.
2. Briand, L., Wüst, J., and Lounis, H. (1999c) "A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: An Industrial Case Study". 21st International Conference on software Engineering, Los Angeles, CA. pp 345-354.
3. Dobing, B. and Parsons, J. (2000) Understanding the Role of Use Cases in UML: A Review and Research Agenda, Journal of Database Management, Vol. 11, No. 4. pp. 28 – 36.
4. Erickson, J., Siau, K. (2003) "Unified Modeling Language? The Good, The Bad, and The Ugly," in: Toppi, H., Brown, C. (eds.), IS Management Handbook, pp483-497, Auerbach.
5. Fenton, N. and Pfleeger, S. (1997) *Software Metrics A Rigorous and Practical Approach,* PWS Publishing, pp. 243-278.
6. Kobryn, C. (2002) "What to Expect from UML 2.0", SD Times, accessed 10/22-2002.
7. Ludwig, B. (1997) "Predicting the Future: Have you considered using the Delphi Methodology?", Extension Journal, October, Vol. 35, No. 5.
8. Miller, G. (1956) "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", *The Psychological Review*, Vol. 63 No. 2, March.
9. Rossi, M. and Brinkkemper, S. (1996) "Complexity Metrics for Systems Development Methods and Techniques," Information Systems, Vol. 21, No. 2 pp. 209-227.
10. Siau, K., and Cao, Q. (2001) "*Unified Modeling Language (UML) – A Complexity Analysis*", Journal of Database Management, Vol. 12, No. 1, pp 26-34.
11. Siau, K., Erickson, J., and Lee, L. (2002) "Complexity of UML: Theoretical versus Practical Complexity", Workshop on Information Technology and Systems (WITS). Barcelona, Spain, December 16-18.
12. Weyuker, E. (1988) "Evaluating Software Complexity Measures", IEEE Transactions on Software Engineering, Vol. 14, No. 9, pp 1357-1365.
13. Zendler, A., Pfeiffer, T., Eicks, M., and Lehner, F. (2001) "Experimental Comparison of Coarse Grained Concepts in UML, OML and TOS", *Journal of Systems and Software,* Vol. 57, pp 21-30.