December 2001

The Complexity of Unified Modeling Language: A GOMS Analysis

Keng Siau
University of Nebraska-Lincoln

Yuhong Tian
University of Nebraska-Lincoln

THE COMPLEXITY OF UNIFIED MODELING LANGUAGE: A GOMS ANALYSIS

Keng Siau
Department of Management
University of Nebraska–Lincoln
ksiau@unl.edu

Yuhong Tian
Department of Management
University of Nebraska–Lincoln

Abstract

Although the unified modeling language (UML) is becoming the de facto *industry standard for object-oriented system development, it is not short of critics. Both researchers and practitioners have agreed that much work needs to be done to enhance UML. This research evaluates the nine UML diagrams using GOMS. GOMS, beginning as a theoretical model in HCI, describes the procedures required for accomplishing a general set of tasks by decomposing the tasks into four elements: **Goals**, **Operators**, **Methods**, and **Selection** rules. We use a special type of GOMS, NGOMSL, for analyzing UML. This research attempts to find ways to enhance the ease of use of UML diagrams and contribute to the evolution of UML.*

Keywords: Unified modeling language, complexity analysis, GOMS.

INTRODUCTION

Unified modeling language (UML) is a visual modeling language tailored specifically for object-oriented design. It can model system requirements, describe designs, and depict implementation details. UML borrows concepts from a large number of different methodologies. Since its inception, UML has emerged as the software industry's dominant modeling language. Supporters argue that UML uses simple, intuitive notations that are understandable to users. It improves the communication by unifying different object-oriented (OO) techniques into one standard and "relieves developers of the proprietary ties that are so common in this industry"(Siau and Cao 2001). On the other hand, diagrams and constructs in UML have been criticized as ambiguous, inadequate, and cognitively misdirected (Dobing and Parsons 2000; Simons and Graham 1999). Hesse (2001) called UML a modern dinosaur and criticized that it "offers a vast variety of competing, complicated and intertwined terms, structures, and relationships." The objective of this research is to understand the complexity of UML and investigate the problems that modelers and users face.

LITERATURE REVIEW

Although UML has been criticized for its complexity, these criticisms are based on gut feelings and "common sense." Systematic and scientific studies to evaluate UML have been lacking. Recently, Siau and Cao (2001) utilized a complexity metric developed by Rossi and Brinkkemper (1996) to analyze UML. Metrics analysis is a systematic quantitative measure. The complexity of the nine UML diagrams are: 26.40 for class diagrams, 10.39 for use case diagrams, 11.18 for activity diagrams, 7.87 for sequence diagrams, 8.12 for collaboration diagrams, 5.92 for object diagrams, 15.39 for state chart diagrams, 15.65 for component diagrams, and 9.95 for deployment diagrams. (N.B. the higher the number, the more complex the diagraming technique). The findings need to be triangulated with empirical evidence.

Siau (2001) proposed the use of a psychological model such as GOMS to evaluate UML. GOMS analysis is an example of applied information processing psychology (Card et al. 1983). It is an engineering model for usability that produces quantitative predictions of how well humans will be able to perform tasks with a proposed interface design (Kieras 1996). GOMS stands for **G**oals, **O**perators, **M**ethods, and **S**election Rules (Card et al. 1983). Goals are what the analyst/user wants to accomplish, operators are actions performed in service of a goal, sequences of operators make up methods, and selection rules are applicable when there is more than one method available.

Several variants of the GOMS analysis technique have been proposed (John and Kieras 1996). The original version, CMN-GOMS, stipulates how to express goals and sub-goals in a hierarchy using methods and operators and how to formulate selection rules. CPM-GOMS assumes that tasks can be performed in parallel, and uses PERT charts to represent the operators and dependencies between operators. NGOMSL (Natural GOMS Language) (Kieras 1988, 1999) presents a procedure to identify all the GOMS components in a form similar to a computer program. NGOMSL can predict the learning time as well as the execution time of using a system. A NGOMSL model includes not only keystroke operators but also internal or mental operators, thus making it possible to be applied in assessing the performance of a complex task such as system modeling.

Although GOMS was initially created to assess the usability of an interface design, the GOMS model has been extended and enhanced over the years and can be applied for analyzing UML. In this research, we will create NGOMSL models for the nine diagramming techniques in UML. The NGOMSL models can then be used to evaluate the different UML diagrams and the results can be triangulated with the complexity metric analysis done by Siau and Cao.

THEORETICAL FOUNDATION

Adaptive control of thought (ACT) (Anderson 1983) is the most popular human-information processing model and is becoming standardized. The ACT production system consists of three memories: working memory (information currently accessible), declarative memory (long-term memory of facts—e.g., definitions of class and use case), and production memory (long-term memory of processes or procedures that operate on facts to solve problems—e.g., the procedure to draw a sequence diagram). The latest version, ACT-R (Anderson and Lebiere 1998), is shown in Figure 1.

There are three memories: Goal Stack encodes the hierarchy of goals, Procedural Memory contains the production rules, and Declarative Memory contains chunks by which declarative knowledge is represented. These three memories are organized by Current Goal, which represents the focus of attention. The Current Goal can retrieve or push goals from or to the Goal Stack. The conflict resolution process selects productions that match the current goal.

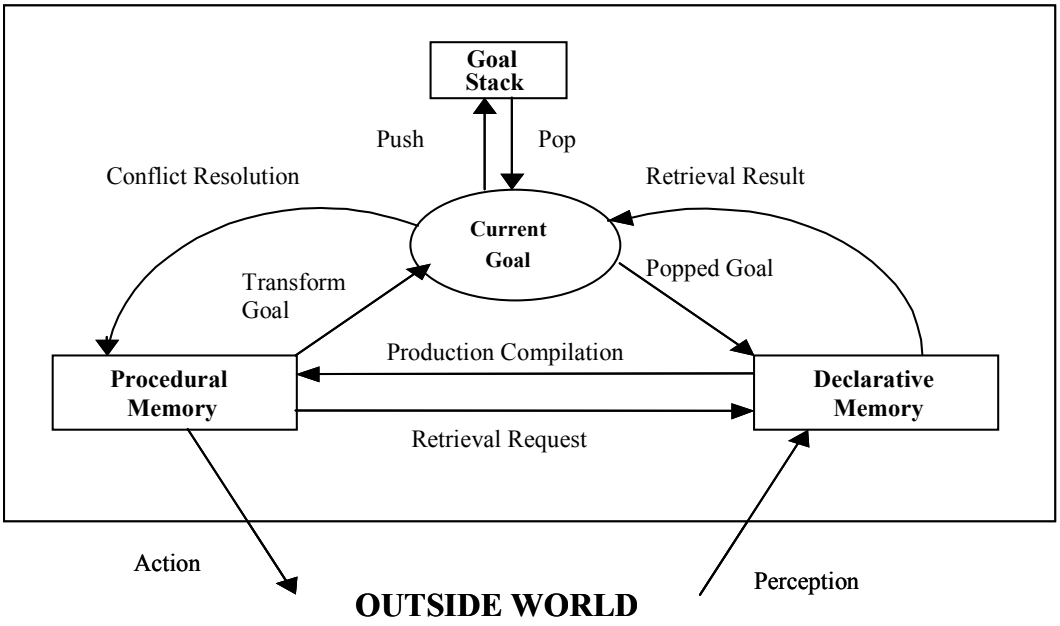

Figure 1. ACT-R

ACT-R serves as a theoretical foundation in this research to support the GOMS model, to guide the construction of the GOMS models for UML, and to help explain the results. The GOMS model we used in this research is NGOMSL (John and Kieras 1996). There are several kinds of operators in NGOMSL. Primary are external operators or observable operators (e.g., draw a class symbol). Another category consists of mental operators (also called M operators). These are non-observable operators (e.g., identify a class or interpret the multiplicity). Kieras (1996) proposed two standard primitive mental operators: one is *flow of control*, which is analogous to CALL-RETURN statements or decide operators (IF-THEN-ELSE). The second is *memory storage and retrieval*, such as RECALL, RETAIN, FORGET, or RETRIEVAL from long-term memory (LTM). The execution times for the second type of mental operators are different between experts and novices. GOMS model builders usually estimate the execution time of the unobservable cognitive operators. Complex mental operators should be bypassed when calculating the execution time. The number of these bypass operators is counted and regarded as an indication of the level of complexity. An example of the NGOMSL model for drawing a use case is presented here:

Method for goal: Draw Use Case

		Operator	Type	Execution Time
1.	Retrieve from LTM the <Use Case definition>	1	(R_LTM)	6 Sec.
2.	Identify a Use Case from the problem domain	1	(SM)	1.35 Sec.
3.	Retrieve from LTM the <Use Case symbol>	1	(R_LTM)	6 Sec.
4.	Draw the Use Case symbol	1	(DR)	3 Sec.
5.	Think-of a <Use Case Name>	1	(SM)	1.35 Sec.
6.	Record the Use Case name	1	(DR)	3 Sec.
7.	Return with accomplished Goal	1		

Note: R_LTM Retrieval from long-term memory operator
SM Standardized M operator
DR Draw Symbol operator

Compared with other variants, the NGOMSL model has more operators that involve internal perceptual and cognitive actions (which are called M operators in NGOMSL). This is important, as M operators are heavily involved in modeling activities. Also, there is an association between statements in NGOMSL and production rules in ACT-R. Thus the number of NGOMSL statements for a modeling technique can be considered as a measure of its cognitive complexity.

RESEARCH QUESTIONS AND PROCEDURE

In this research, the complexity of UML diagrams is measured by: (1) number of statements or operators; (2) time of execution; (3) layers of goal structure; (4) number of bypassed operators.

Research Procedure

We will use protocol analysis to create the NGOMSL model for UML diagrams. Because the NGOMSL model for modeling is different from that of interpreting a model, we will create separate NGOMSL models for modeling and interpretation. The procedure is as follows:

Step 1: Model Building

- UML experts will be exposed to several real-world modeling tasks. They have to "think aloud" while modeling. The whole process is video-recorded. For the interpretation task, UML novices will be used.
- A different group of UML experts will be trained in the NGOMSL model. They will generate the NGOMSL models (for both modeling and interpretation) for the nine UML diagramming techniques based on their expertise and the subjects' transcripts.
- Two standardized NGOMSL models (i.e., modeling and interpretation) will be created for each UML diagramming technique giving us a total of 18 NGOMSL models.
- Compute the execution time of each operator from the video recordings or use standard estimations from prior research.
- Check the sensitivity of judgment calls and assumptions. In NGOMSL, this involves checking whether the performance estimation will change drastically if the judgment or assumption is different.

Step 2: Evaluation
- Count the NGOMSL statements (operators).
- Calculate execution time of NGOMSL statements.
- Count the layers of the goal structure.
- Count the number of bypassed operators.

Step 3: Validation and Triangulation Study
- Triangulate the results obtained from the NGOMSL analysis to the metrics analysis results in Siau and Cao (2001)

Limitations

This research is limited in two aspects. First, measuring the execution time of the M operators is difficult. Second, as in all GOMS models, the level of granularity is difficult to control. NGOMSL, fortunately, provides some systematic ways to control these issues. Using bypassed operators, which can be used as an indication of the complexity of the NGOMSL model, alleviates the first problem. For the second issue, the research procedure can relieve some of the problems. Since there will be more than one subject involved in the model building, the level of details is normalized by synthesizing the various models created by individuals.

PRELIMINARY RESULTS

As a pilot study, we carried out a modeling task analysis of class diagram and use case diagram and built a NGOMSL model for each of them. The execution time of each operator type is evaluated based on the NGOMSL model's recommended time. As the NGOMSL model suggests, we use 1.35 seconds for the standardized M (SM) operator. For long-term memory retrieval (R_LTM), we use the recommended value of 6 sec/chunk (Kieras 1996, pg. 41). The time for storing information into LTM (S_LTM) is set by Card et al. (1983, Chapter 2) to be 10 sec/chunk. Drawing a construct or recording a name (DR) is assumed to be three seconds. For complicated decide operators (e.g., identifying association type), we marked them as bypassed operators (BP). In other words, we did not compute the execution time for complicated decide operators but we added the total number of bypassed operators and compared the numbers among the diagrams in the final analysis.

The preliminary results show that there are four layers of goals and 51 operators in drawing/modeling a class diagram. The total execution time for class diagrams is 118.5 seconds with one bypass operator. For modeling a use case diagram, the goal structure has three layers. There are 29 operators. The total execution time is 66.75 seconds with one bypass operator.

The preliminary results using NGOMSL analysis show that the use case diagram is easier to model than the class diagram in that it takes less time, has a smaller statement count, and has fewer layers of goal structure. The number of bypass operators for modeling a use case diagram is the same as those of modeling a class diagram. The relative complexity of these two diagrams is similar to the results obtained by Siau and Cao (2001). Their complexity analysis of UML revealed that the complexity index for the class diagram (26.4) is twice that for the use case diagram (10.39). This triangulation provides a form of validity check.

CONCLUSION

This is the first research that applies the GOMS model to evaluate modeling methods. Further work will be carried out to build the two sets (i.e., modeling and interpretation) of NGOMSL models for the nine UML diagrams. Detailed analysis of these NGOMSL models will bring about enriched understanding regarding the complexity and usability of the nine UML diagrams.

Rerferences

Anderson, J. R. *The Architecture of Cognition*, Harvard University Press, Cambridge, MA, 1983.
Anderson, J. R., and Lebiere, C. *The Atomic Components of Thought*, Lawrence Erlbaum Associates, Mahwah, NJ, 1998.
Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Mahwah, NJ, 1983.

Dobing, B., and Parsons, J. "Understanding the Role of Use Cases in UML: A Review and Research Agenda," *Journal of Database Management* (11:4), 2000, pp. 28-36.

Hesse, W. "RUP: A Process Model for Working with UML?," in *Unified Modeling Language: Systems Analysis, Design and Development Issues*, K. Siau and T. Halpin (eds.), Idea Group Publishing, Hershey, PA, 2001, pp. 64-74.

John, B. E., and Kieras, D. E. "The GOMS Family of Analysis Techniques: Comparison and Contrast," *ACM Transactions on Computer-Human Interaction* (3:4), 1996, pp. 320-351.

Kieras, D. E. "A Guide to GOMS Model Usability Evaluation using GOMSL and GLENA3," 1999 (downloaded from http://www.eecs.umich.edu/~kieras/goms.html).

Kieras, D. E. "Towards a Practical GOMS Model Methodology for User Interface Design," in *The Handbook of Human-Computer Interaction*, M. Helander (ed.), North Holland, Amsterdam, 1988, pp. 135-158.

Kieras, D. E. "A Guide to GOMS Model Usability Evaluation Using NGOMSL," 1996 (downloaded from http://www.eecs.umich.edu/~kieras/goms.html).

Rossi, M., and Brinkkemper, S. "Complexity Metrics for System Development Methods and Techniques," *Information Systems* (21:2), 1996, pp. 209-227.

Siau, K. "Rational Unified Process and Unified Modeling Language: A GOMS Analysis," in *Unified Modeling Language: Systems Analysis, Design and Development Issues*, K. Siau and T. Halpin (eds.), Idea Group Publishing, Hershey, PA, 2001, pp. 107-116.

Siau, K., and Cao, Q. "Unified Modeling Language (UML): A Complexity Analysis," *Journal of Database Management* (11:1), 2001, pp. 26-34.

Simons, A. J. H. ,and Graham, I. "Thirty Things That Go Wrong in Object Modeling with UML 1.3," in *Precise Behavioral Specification of Businesses and Systems*, H. Kilov, B. Rumpe, and I. Simmonds (eds.), Kluwer Academic Publishers, Boston, 1999.