IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA ITB7417
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

Chapter XV

How Complex Is the Unified Modeling Language?

Keng Siau
University of Nebraska–Lincoln, USA

Qing Cao
University of Missouri–Kansas City, USA

Unified Modeling Language (UML) has emerged as the software industry's dominant modeling language. It is the *de facto* modeling language standard for specifying, visualizing, constructing, and documenting the components of software systems. Despite its prominence and status as the standard modeling language, UML has its critics. Opponents argue that it is complex and difficult to learn. Some question the rationale of having nine diagramming techniques in UML and the raison d'être of those nine techniques in UML. Others point out that UML lacks a comprehensive methodology to guide its users, which makes the language even more convoluted. A few studies on UML can be found in the literature. However, no study exists to provide a quantitative measure of UML complexity or to compare UML with other object-oriented techniques. In this research, we evaluate the complexity of UML using complexity metrics. The objective is to provide a reliable and accurate quantitative measure of UML complexity. A comparison of the complexity metrical values of UML with other object-oriented techniques was also carried out. Our findings suggest that each diagram in UML is not distinctly more complex than techniques in other modeling methods. But as a whole, UML is very complex–2-11 times more complex than other modeling methods.

INTRODUCTION

Unified Modeling Language (UML) is a visual modeling language for modeling system requirements, describing designs, and depicting implementation details. Grady Booch, Jim Rumbaugh, and Ivars Jacobson, known collectively as the "three Amigos" at Rational Software Corp, spearheaded development of UML in the mid-1990s. Unified Modeling Language (UML) borrows concepts from a large number of different methods, and is tailored specifically for object-oriented systems development. Soon after its

This chapter appears in the book, Advanced Topics in Database Research by Keng Siau.

inception, UML emerged as the software industry's dominant modeling language. UML is not only the *de facto* modeling language standard for specifying, visualizing, constructing, and documenting the components of software systems, but it has also been accepted by the Object Management Group (OMG) as a standard language for object-oriented analysis and design. In addition, UML has been proposed for standardization by the International Standards Organization (ISO) and approval is anticipated sometime in 2001.

Many of the language's supporters claim that UML's simplicity is its chief benefit (Kobryn, 1999) and argue that UML uses simple, intuitive notations that are understandable by nonprogrammers. This is in line with a recent study by Fedorowicz and Villeneuve (1999) who surveyed vendors and developers regarding OO systems analysis, design, and programming. Their results suggested that the OO approach is intuitive and easy to use/comprehend for vendors and developers alike. By offering a common blueprint language, UML also relieves developers of the proprietary ties that are so common in this industry. Major vendors including IBM, Microsoft, and Oracle are brought together under the UML umbrella. If developers, customers, and implementers can all understand a single modeling language instead of the few dozen OO techniques that existed before UML (Siau, 1999), they are more likely to agree on the intended functionality of the modeling techniques and constructs, thereby improving the communication process among the stakeholders and enhancing their chances of creating an application that truly addresses business problems.

UML, nevertheless, has its fair share of critics. Halpin and Bloesch (1999) pointed out that although UML facilitates the transition to object-oriented code, its implementation concerns render it less suitable for developing and validating a conceptual model with domain experts. Beckwith and Moore (1998) studied the use of deployment diagrams in architecture modeling and discovered problems with multi-tasking. In contrast to Kobryn (1999) who stated that UML is simple to use, Douglass (1998) argued that UML is somewhat large and complex, which can be daunting to novice users. He called for future research to examine the questions: (1) How easy is it to understand UML? and (2) How easy is it to use UML?

This study investigates the complexity of UML. Complexity is a key measure of the effectiveness of a language because complexity directly affects the learn-ability and ease-of-use of the language. As mentioned by Booch, Rumbaugh, and Jacobson (1999a), UML is still in its infancy and there is a need for improvement in the analysis of UML and in understanding its use and functionality.

The rest of the chapter is organized as follows: the following section reviews UML and is followed by a section discussing various approaches to measuring complexity, including empirical studies and different complexity metrics. The complexity of UML is then analyzed using the complexity metrics proposed by Rossi and Brinkkemper (1994). The research findings are then discussed and a comparison is then made between the complexity of UML and other modeling techniques. The last section summarizes the research, lists the limitations of the study, and suggests future research directions.

OBJECT ORIENTATION AND UML

Object-orientation (OO) is the new paradigm for systems development (Johnson & Hardgrave, 1999). The first few years of the 1990s saw the blossoming of nearly 50 different object oriented methods. The abundance of OO methods caused great confusion among users. Standardization of OO was needed to curtail the chaos and the Unified Modeling Language (UML) was the result.