EDITORIAL PREFACE

An Analysis of Unified Modeling Language (UML) Graphical Constructs Based on BWW Ontology

Keng Siau, University of Nebraska-Lincoln, USA

ABSTRACT

Prior research on system analysis and design indicates that graphical constructs have a significant impact on interpreting information conveyed in diagrams. This article examines Unified Modeling Language (UML) graphical constructs. Evaluation of UML graphical constructs is important as they are the building blocks of UML diagrams. In this paper, graphical constructs used in UML are discussed and issues with UML graphical constructs are identified. The paper also suggests ways to alleviate these issues.

Keywords: BWW Ontology; Graphical Constructs; Ontological Discrepancies; Unified Modeling Language

1.0 INTRODUCTION

The Unified Modeling Language (UML) is a visual modeling language dominant in object-oriented software development. This language was adopted by the Object Management Group (OMG) as its standard modeling language in 1997. UML 1.x defines nine diagramming techniques: Class diagram, Object diagram, Component diagram, Deployment diagram, Use Case diagram, State-chart diagram, Activity diagram, Sequence diagram, and Collaboration diagram (Booch et al., 1999). UML 2.x defines thirteen diagramming techniques: Class Diagram, Object diagram, Component diagram, Composite Structure diagram, Package diagram, Deployment diagram, Use Case diagram, Sequence diagram, Communication diagram, State diagram, Activity diagram, Timing diagram, and Interaction Overview diagram (Booch et al., 2005). These diagrams are interrelated and depict different aspects of the problem domain.

Since its introduction, UML has been a popular research topic (e.g., Siau & Cao, 2001; Siau & Tan, 2006; Gemino & Parker, 2009; Siau & Tian, 2009; VanderMeer & Dutta, 2009). Many of these research studies evaluate the usability of UML (e.g., Siau & Lee 2004, Siau & Wang, 2007; Reinhartz-Berger & Sturm, 2008). Evaluation studies on UML have pointed to the difficulty in learning UML
As a visual modeling language, UML relies heavily on graphical constructs. These graphical constructs form the foundation of the various UML diagrams. For example, UML uses a rectangle to represent classes in Class Diagram, an ellipse to describe use cases in Use Case Diagram, an arrow to indicate direction of messages in Sequence Diagram (see Figure 1).

Diagrams are useful in a number of ways. First, diagrams can group relevant data together, thus minimizing the time spent on searching for required elements. Second, diagrams can group information about a single element together, thus avoiding the need to match labels. Third, diagrams support a large amount of perceptual inference cues, which are easy for humans to use. In short, diagrammatic representations have positive effects on the three components of human information processing: search, recognition, and inference (Larkin & Simon, 1987).

Nevertheless, the same information may be represented in different ways, but different representations of the same information may not be equally easy to use (Larkin & Simon, 1987). In other words, diagrams that are equivalent based on information structure, but they might be not computationally equivalent because different diagrammatic representations may facilitate search, recognition, and inference to varying degrees (see Figure 2) (Siau, 2004).

This paper analyzes and evaluates the UML graphical constructs and identifies some issues with UML graphical constructs. At the end of the paper, suggestions are made to alleviate the problems. The research analyzes and evaluates the UML graphical constructs using BWW Ontology (Wand & Weber, 1993).

2.0 ONTOLOGICAL DISCREPANCIES

Ontology is a “philosophical theory concerning the basic traits of the world” (Bunge 1977: p.38). As a meta-theory of the structure and behavior of real world systems, ontology deals with the nature of things in general as opposed to particular things (Weber 2003; Evermann & Wand, 2009). Wand and Weber (1993) identified four possible ontological discrepancies in information system grammar.

1. Construct deficiency occurs when an ontological construct is not represented by any grammar constructs. This implies that the grammar constructs are not complete and thus may cause the problem that certain aspects of the real system cannot be captured.

2. Construct overload occurs when one grammar construct is used to represent more than one ontological construct. This means that the same grammar construct can be interpreted in different ways, and thus may result in difficulties in interpreting models.

3. Construct redundancy occurs when more than one grammar construct is used to represent one ontological construct. This may make the modeling grammar overly complex.

4. Construct excess occurs when one or more grammar constructs do not represent any ontological constructs at all. This means an excess of grammar constructs and may
also result in an overly complex modeling grammar.

3.0 ISSUES WITH UML GRAPHICAL CONSTRUCTS

UML graphical constructs serve as the building blocks of UML diagrams.

Previous research on the data model in system analysis and design shows that pragmatic features of diagrammatic representation significantly influence model interpretation (Nordbotten & Crosby, 1999). In other words, the details of the graphic constructs are of significant importance. Despite its status as the de facto modeling language for object-oriented systems development, UML is still in its infancy and is still evolving. Research on UML and the evaluation of UML notational elements is of great importance. To date, however, there has been little focus on the studying of UML graphical constructs. This article examines UML graphical constructs. Specifically, the paper discusses three issues related to UML graphical constructs.

First, some UML notational elements are not differentiated enough. Some UML graphical constructs are too similar. UML defines
different line styles (e.g., dashed lines and solid lines), arrowhead styles (stick arrowhead, empty arrowhead and filled arrowhead), and various combinations of lines and arrowhead styles (see Figure 3). For example, solid lines with different arrowhead styles can represent different messages, such as a sequential message, an asynchronous message, and a call. Empty arrowheads with different line styles are used to represent generalization and realization. The similarity in graphical constructs in UML may hinder the human information cognitive process of search, recognition, and inference, and thus, may make the process of linking relevant graphical constructs and making inference difficult.

Second, construct overload is found in UML graphical constructs. Based on the definitions given by Wand and Weber (1993), a construct overload is when one UML graphical construct can represent two or more UML concepts. Many two-dimensional symbols are used in more than one diagram (see Figure 4). For example, the rectangle is used in Class diagram and Object diagram to represent classes and objects respectively. A rounded rectangle is used to represent activity state and action state in the activity diagram. An ellipse is used for base use case, extended use case, and including use case. Construct overload in UML means that some graphical constructs can be interpreted in different ways, and thus may result in confusion or misinterpretation.

Third, construct redundancy is also present in UML notational elements. Based on definitions by Wand and Weber (1993), a construct redundancy in UML is when two or more UML graphical constructs can represent the same UML concept (see Figure 5). For example, icons such as a computer and a cube are used to represent a server in a Deployment Diagram. 2-dimensional symbols such as the rectangle and the circle can be used to represent an interface. Generalization can be represented by two style direct and tree style. UML was found to be two to 11 times more complex than other

Figure 3. Examples of similarity in graphical constructs

<table>
<thead>
<tr>
<th>Arrowheads</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>➞ ➞</td>
<td>———</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>generalization</td>
<td>realization</td>
</tr>
<tr>
<td>——— ➞</td>
<td>——— ➞</td>
</tr>
<tr>
<td>sequential message</td>
<td>asynchronous message</td>
</tr>
</tbody>
</table>
object-oriented methods (Siau & Cao, 2001). Construct redundancy may make UML more complex than necessary.

4.0 DISCUSSIONS AND RECOMMENDATIONS

There is a need to re-think and re-design a few of the UML graphical constructs. In addition to the use of lines and arrowheads to represent various relationships, we propose other visual cues, such as color, be employed in UML to better represent relationships.

Another suggestion is the elimination of construct redundancy in UML graphical constructs. The use of more than one UML graphical constructs to represent one concept is genuinely redundant, and causes unnecessary confusion to UML users. It indicates a weakness that should be ameliorated in future versions of UML. The standardization of those inconsistent UML graphical constructs is necessary.

Construct overload in UML graphical constructs is troublesome, but may not be a critical problem. Although a two-dimensional symbol may be used for different concepts, it is usually used in combination with other UML graphical constructs. For example, a rectangle is used for both Class and Object, but it always comes with a text string that names the class or object. The surface semantics (Siau et al., 1997) and the different naming formats for class or object help differentiate the Class from Object and vice versa. An ellipse is used for Base Use Case, Including Use Case, and Extended Use Case. But when use cases are placed in a Use Case Diagram, the extend relationship and include relationship distinguishes the base use case from the extended and including use cases. Considering that the overloaded UML graphical constructs are used in conjunction with other constructs or complemented with text strings, it is probably unnecessary to add more graphical constructs to UML. Although adding more graphical constructs to UML to eliminate the construct overload issue is ideal, an increase in two-dimensional symbols might make UML harder to learn than it already is. Human’s cognitive limitations need to be taken into account (Siau, 1999). Further, it is hard to come up with unique two-dimensional symbols to represent different graphical constructs.

One way to reduce confusion among UML graphical constructs is to identify the core constructs and diagrams in UML. These small set of core constructs (typically around 20% using the 80-20 rule) should be clear and

Figure 4. Examples of construct overload in UML

![Diagram showing examples of construct overload in UML](image-url)
unambiguous as they will account for 80% of the UML usage. Effort to reduce the ontological deficiencies of these core UML graphical constructs will go a long way towards reducing confusion in interpreting UML diagrams.

Another point to note is the theoretical and practical complexity (Siau et al., 2005). UML may be theoretically very complex when all the graphical constructs and diagrams are taken into account. But the 80-20 rule will argue that only 20% of the graphical constructs will be used 80% of the time. Thus, practically, UML’s complexity may be lower.

5.0 CONCLUSION

UML is gaining widespread acceptance. The emergence of UML represents a major milestone in the information systems field as the researchers and practitioners can now focus on improving and enhancing one modeling language instead of a few dozen modeling languages (Siau, 2007). Nevertheless, UML still has many rough edges to smooth out. More research on UML and usability studies (see Siau & Tan, 2005) to evaluate UML are needed. The design of UML graphical constructs is one area that needs much research attention. This paper identifies issues in UML graphical constructs and makes recommendations to alleviate those problems. Changes to UML graphical constructs should be made quickly and incorporated into new releases of UML. It will be harder and more costly to rectify the problems with UML graphical constructs as UML becomes more widely used and institutionalized.

REFERENCES


