Chapter 13
Dimensions of UML
Diagram Use:
Practitioner Survey and Research Agenda

Brian Dobing
University of Lethbridge, Canada

Jeffrey Parsons
Memorial University of Newfoundland, Canada

ABSTRACT
The Unified Modeling Language (UML) is an industry standard for object-oriented software engineering. However, there is little empirical evidence on how the UML is used. This chapter reports results of a survey of UML practitioners. The authors found differences in several dimensions of UML diagram usage on software development projects, including frequency, the purposes for which they were used, and the roles of clients/users in their creation and approval. System developers are often ignoring the “Use Case-driven” prescription that permeates much of the UML literature, making limited or no use of either Use Case Diagrams or textual Use Case descriptions. Implications and areas requiring further investigation are discussed.

INTRODUCTION
The Unified Modeling Language (UML) emerged in the mid-1990s through the combination of previously competing Object-Oriented Analysis and Design (OOAD) approaches (Rumbaugh, Blaha, Premerlani, Eddy, and Lorensen, 1991; Jacobson, Christerson, Jonsson, and Overgaard, 1992; Booch, 1994), along with other contributions to modeling complex systems (e.g., Harel, 1987). Control over its formal evolution was placed in the hands of the Object Management Group (www.omg.org), which oversaw a major revision to Version 2 in 2006 (Selic, 2006) and recently released the UML 2.2 (Object Management Group, 2009). The UML became widely accepted as the standard for OOAD soon after its introduction (Kobryn, 1999) and remains so today (Evermann and Wand, 2006). A large number of practitioner articles and dozens of textbooks have been devoted to articulating various aspects of the language, including guidelines for using it. More recently, a substantial body of research on the UML has emerged, including ontological analysis of its modeling constructs (Evermann and Wand,
2001a, 2001b) and a more recent empirical assessment (Evermann and Wand, 2006), analysis of the language’s complexity (Siau and Cao, 2001, 2002; Erickson and Siau, 2007), related learning difficulties (Siau and Loo, 2006) and means to address them (Batra and Satzinger, 2006), and experiments that evaluate various aspects of the effectiveness of UML models (Burton-Jones and Weber, 2003, Burton-Jones and Meso, 2006). Batra (2008, p.i) also lists a number of recent UML research areas.

The UML was not developed based on any theoretical principles regarding the constructs required for an effective and usable modeling language for analysis and design; instead, it arose from (sometimes conflicting) “best practices” in parts of the software engineering community (Booch, 1999; Booch, Rumbaugh, and Jacobson, 1999). This resulted in a language containing many modeling constructs, which has thus been criticized on the grounds that it is excessively complex (Dori, 2002; Kobryn, 2002; DeJong, 2006). However, more recently research has suggested the “practical complexity” is not as great (Siau, Erickson and Lee, 2005; Erickson and Siau, 2007). At the same time, the UML has also been criticized for lacking the flexibility to handle certain modeling requirements in specific domains (Duddy, 2002). As a consequence, the UML has evolved to allow for the definition of “profiles” that have enabled Domain Specific Languages (Cook, 2000; DeJong, 2006).

While the UML is intended to be “largely process-independent,” some of the key originators recommend a Use Case-driven process (e.g., Booch et al., 1999, p.33). A majority of UML books since then have endorsed this view, and most contain at least some further prescriptions for applying the language in modeling (Stevens and Pooley, 2000; Schneider and Winters, 2001; Larman, 2005). As would be expected with a best practices approach, their prescriptions sometimes differ. While some accept the original view that only Use Case Narratives (or, more simply, Use Cases) be used to verify requirements with users (Jacobson, Ericsson, and Jacobson, 1994), others explicitly or implicitly indicate that other UML diagrams can be used for this purpose, e.g., Activity Diagrams “can be safely shared with customers, even those unfamiliar with software engineering” (Schneider and Winters, 2001, p.67).

There are also differences in guidelines for using the language, and Use Case Narratives in particular (Dobing and Parsons, 2000). This is not surprising since the official UML documentation (currently 2.2) has never provided guidance on Narrative format, stating only that “use cases are typically specified in various idiosyncratic formats such as natural language, tables, trees, etc.” (Object Management Group, 2009, p.592). However, there is no shortage of information on Use Cases. As of November 2009, Amazon.com lists nine books with “Use Case” in the title (related to system modeling), but none with “Class Diagram” (although there are many UML books covering both). Finally, when the Use Case-driven approach is used, concerns have been raised about the potential communication disconnect (Dobing and Parsons, 2000) that can occur when Use Cases are the primary communication tool among analysts and the clients/users on the project team while Class Diagrams play that role among analysts and programmers. While Use Case Narratives have been found to be the most comprehensible artifact for managers, users and domain experts, and even more so when used with Use Case Diagrams (Gemino and Parker, 2008), they are the least comprehensible for designers and programmers (Arlow and Neustadt, 2004) when they require knowledge of the organizational context that programmers do not have. Conversely, Class Diagrams are highly comprehensible by programmers, but not clients/users (Arlow and Neustadt, 2004).

In view of these issues, it would not be surprising to find a variety of practices followed by UML practitioners. We believe understanding current practice can make an important contribution to