Unified Modeling Language 2.0

Peter Fettke
Institute for Information Systems (IWi) at the DFKI, Germany

INTRODUCTION

Mature engineering disciplines are generally characterized by accepted methodical standards for describing all relevant artifacts of their subject matter. Such standards not only enable practitioners to collaborate, but they also contribute to the development of the whole discipline. In 1994, Grady Booch, Jim Rumbaugh, and Ivar Jacobson joined together to unify the plethora of existing object-oriented systems engineering approaches at semantic and notation level (Booch, 2002; Fowler, 2004; Rumbaugh, Jacobson & Booch, 1998). Their effort leads to the unified modeling language (UML), a well-known, general-purpose, tool-supported, process-independent, and industry-standardized modeling language for visualizing, describing, specifying, and documenting systems artifacts.

UML is applicable to software and non-software domains, including software architecture (Medvidovic, Rosenblum, Redmiles, & Robbins, 2002), real-time and embedded systems (Douglass, 2004), business applications (Eriksson & Penker, 2000), manufacturing systems (Bruccoleri, Dieaga, & Perrone, 2003), electronic commerce systems (Saleh, 2002), data warehousing (Dolk, 2000), bioinformatics (Bornberg-Bauer & Paton, 2002) and others. The language uses multiple views to specify system's structure and behavior. Modeling tools supporting the development of UML diagrams are available from a number of commercial vendors and the open source community (OMG, 2006b; Robbins & Redmiles, 2000).

Table 1 depicts the origin and descent of UML. The recent version UML 2.0 supports thirteen different diagram types. Table 2 overviews the main concepts of each diagram, a more detailed description is given below. For a full description of all semantics see (OMG, 2005a, 2005b, 2006a, 2006c) respectively the available secondary literature (Fowler, 2004; Rumbaugh et al., 1998).

UML version 2, first planned for 2001 (Kobryn, 1999, p. 30), was finally completed in 2006. This major revision mainly focuses on language extensibility, language specification, language precision and expressiveness. Although the complete language specification was almost fully rewritten, this revision is primary an internal reorganization with just minor consequences for the end user. For example, the new diagrams mainly clarify and resemble existing diagram types.

The description of UML 2.0 consists of four separate documents (Kobryn, 2002):

- **Infrastructure:** This document is concerned with core language features. It specifies the base classes that provide the foundation for UML modeling concepts.
- **Superstructure:** Advanced topics such as component and activity modeling are defined by this specification. It describes the constructs that developers use to build UML models.
- **Object constraint language (OCL):** This specification describes the language used for invariants, operation specifications etc.

Table 1. History of UML (Fowler, 2004, pp. 151-159; Kobryn, 1999, p. 30)

Year	Version	Comments
1995	0.8	Origin of UML, so-called "Unified Method"
1996	0.9	Refined proposal
1997	1.0	Initial submission to OMG
1997	1.1	Final submission to OMG
1998	1.2	Editorial revision with no significant technical changes
1999	1.3	New use case relationships, revised activity diagram semantics
2001	1.4	Minor revisions, addition of profiles
2003	1.5	Adding action semantics
2005	1.4.2	Standardized by the International Organization for Standardization (ISO/IEC 19501:2005)
2005/6	2.0	Deep changes to meta-model, new diagram types, improved expressiveness

Table 2. UML diagram types

Focus	Diagram	Purpose	Main Concepts	Supported Since
Structure diagrams	Class	Object structure	Class, features, relationships	UML 1
	Object	Example configuration of instances	Object, link	UML 1 (unofficially)
	Component	Structure and connections of components	Component, interface, dependency	UML 1
	Composite structure	Decomposition of a class during runtime	Part, interface, connector, port	UML 2
	Package	Interrelationships between packages	Package, dependency	UML 1 (unofficially)
	Deployment	Deployment of components to nodes	Node, component, dependency	UML 1
Behavior diagrams	Use case	User interaction with system	Use case, actor	UML 1
	Activity	Procedural and parallel behavior	State, activity, completion, transition, fork, join	UML 1
	State machine	Change of events during object's lifetime	State, transition, event, action	UML 1 statechart diagram
Interaction diagrams	Sequence	Interaction between objects emphasizing sequences	Interaction, message	UML 1
	Communication	Interaction between objects emphasizing collaborations	Collaboration, interaction, message	UML 1 collaboration diagram
	Timing	Interaction between objects emphasizing timings	Object, timing constraint, state, event	UML 2
	Interaction	Interplay between activities and sequence interactions	Combination of sequence and activity diagram (see there)	UML 2

- **Diagram interchange:** The storage and exchange of model including the layout of UML models is covered by this specification.

The specification of the UML is publicly available and maintained by the Object Management Group (OMG). OMG's standardization process is formalized and consists of several proposal, revision, and final implementation activities (Kobryn, 1999, p. 31f.). Note, UML 1.4.2 is adopted by the International Organization for Standardization, too.

BACKGROUND

There is a great deal of terminological confusion in the modeling literature. A modeling language or grammar provides a set of constructs and rules that specify how to combine the constructs to model a system (Wand & Weber, 2002, p. 364). It can be distinguished between an abstract syntax and a concrete syntax or notation of a language. While the abstract syntax specifies conceptual relationships between the constructs of the language, the concrete notation defines symbols representing the abstract constructs. In contrast, a modeling method provides procedures by which a language can be used. A consistent and suited set of modeling methods is called a methodology. A model is a description of a domain using a particular modeling language.

The UML specification provides an abstract syntax and a concrete notation for all UML diagrams as well as an informal description of the constructs' semantics. The UML's language specification is independent of but strongly related to other OMG standards such as Common Data Warehouse Model, XML Metadata Interchange or Meta Object Facility. A modeling method or a modeling methodology is not defined by the UML standard. Hence, the language is process-neutral and can be used with different software development processes.