The Evolution of UML

D

Rebecca Platt
Murdoch University, Australia

Nik Thompson
Murdoch University, Australia

INTRODUCTION

Since its inception, the Unified Modeling Language (UML) has risen to relative ubiquity in the IT community. However, despite its status as an ISO industry standard (International Organization for Standardization, 2005), the UML is still evolving to accommodate the changing needs of industry. This development aims to ensure that UML remains effective and relevant to the most current developments in software engineering techniques. This article charts the progress of this arguably indispensable standard and discusses the ongoing evolution in three sections: The Past, The Present, and The Future. The Past section will detail the reasons for which standardization was needed, the history behind its inception and development, initial reception from the user community and also its initial effectiveness. The Present section then describes the various changes between UML 1.0 and UML 2.4.1. The reasons behind these changes and the effectiveness of them are then discussed. Finally in The Future section, the article will describe the current state of UML, predictions for the next specification of UML based on the Object Management Group documentation, and also common problems and suggestions from the wider community which may be addressed in future iterations of the specification.

BACKGROUND

The Unified Modeling Language is a form of notation that was developed with the core goal of creating a standardized representation of general-purpose models, with the focus of functionality of these primarily being for software engineering and systems development. Despite this main focus of approach in the specification design, the language is meant to attain some level of applicability regardless of the subject matter. The reason a modeling language was needed in order to achieve this was to manage the complexity of the subject at hand - whether it was system or software design or another subject entirely. As a model is by nature an abstraction of reality, it allows the user to characterize the design of the subject in an effective manner. This abstract model then enables the user to better evaluate the subject and communicate that in an efficient and meaningful way rather than attempting to demonstrate their intentions using the actual software or system in question. In order to achieve this intended core goal the language has been modified and refined over time, resulting in evolutions of varying effectiveness and popularity.

THE EVOLUTION OF UML

The Past

In the late 1950s, the first object orientated programming language, Simula was introduced, and with it came "a powerful new combination of ideas into structuring computer programs, including instantiation of abstract data types, inheritance, and polymorphism" (Cook, 2012, p. 471). To accompany this new idea of object orientated languages, methods for designing software in this object orientated way also started to emerge, and in time they were referred to as modeling languages. By the late 1980s there were more than fifty separate modeling languages - each with their own syntax, structure and notation. There were many issues with this overwhelming variety of languages and it has been noted that "such open-ended approaches [could] affect and constrain the system in unexpected ways or even

DOI: 10.4018/978-1-4666-5888-2.ch186

result in failure. For example, system development and implementation failure rates remained stubbornly high. Cost overruns and time overruns were still the norm, rather than the exception" (Erickson & Siau, 2013, p. 296). As it was humanly impossible in this kind of environment for all system analysts and other relevant personnel to be trained in all methods, the lack of communication and technical understanding coupled with the fact that the majority of the languages available were unable to meet the demands required of them, led to alarmingly high project failure rates.

This lack of standardization and communication was not only negatively affecting development projects but also limiting the potential of object-orientated technology in general. In response to this very significant concern, The Object Management Group (OMG) was founded in 1989. The initial and presiding goal of OMG was to "create a standard for communication amongst distributed objects" (Cook, 2012, p. 472). This goal was intended to foster progress toward a common object model that would work on all platforms on all kinds of development projects. In order to further this goal specifically in the domain of modeling languages, OMG launched the Object Analysis and Design Special Interest Group to study design methods. This is also the origin point from which any Request For Proposals were issued.

Around the time that OMG was founded, a separate company called Rational was also attempting to implement a solution to the over saturation of modeling languages in use. To this end they recruited Grady Booch and James Rumbaugh in 1996. These men were the creators of two of the dominant modeling languages of the time. Booch's method was called Object-Oriented Design (OOD) (Booch, 1991) and Rumbaugh's method was known as the Object-Modeling Technique (OMT) (Rumbaugh, Blaha, Lorensen, Eddy, & Premerlani, 1990). They were soon joined by Ivar Jacobson, whose Object-Oriented Software Engineering (OOSE) method (Jacobson, 1992) was also a prominent modeling language at the time. "The Three Amigos" as they later came to be known then set to work on the development of the Unified Modeling Language. A potentially universal standard form of notation with the intent to create ease of communication and reduce the risk of failure for projects, with human factors considered above all as this had been identified as a main failure point of previous projects (Erickson & Siau, 2013).

The UML 0.91 specification was the initial result of the unification of OOD, OMT, and OOSE, a somewhat successful endeavor as each base modeling language had unique strengths; Booch's OOD was good for low level design, Rumbaugh's OMT was effective for OO analysis, and Jacobson's OOSE was good for high level design, as well as allowing for the implementation of use cases. Working with "The Three Amigos" were the UML Partners; a software development team who represented a range of different of vendors and system integrators, who would collaborate to propose UML as the standard modeling language for the OMG (Kobryn, 1999). Representatives from other companies (such as IBM, Microsoft and Oracle) were consulted during the Object-Oriented Programming, Systems, Languages and Applications (OOPSLA) conference held that year, with the outcome of these consultations resulting in the UML 1.0 draft which was then submitted to OMG in response to the Request For Proposal. UML 1.0 was accepted by OMG in November, 1997.

The initial response after the release of the specification indicated that the Unified Modeling Language was very effective, once the personnel involved had made it past the difficult learning curve of training in a new modeling language. In fact there is speculation that the response towards UML was actually too great - for while it was proven to be much more effective than its predecessors, it still had issues. The rapid uptake and positive response meant that the uptake of UML ended up growing at an alarming rate before it had finished standardizing properly.

The Present

When initially accepted as a standard, UML 1.0 appeared to meet all stated requirements and to be an effective modeling language. Since then, however, a number of revisions have taken place to alter the notation in order to fix various shortcomings and to become more effective. For example, some of the issues that were resolved between UML 1.1 and UML 1.3 included the lack of integration between certain model types, the absence of certain modelers and that some of the standard elements were named and organized inconsistently. There was also trouble with the architectural alignment – According to OMG "The submitters fell short of their goal of implementing a 4-layer metamodel architecture using a strict metamod-