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 chapter charts the progress of this arguably indispensable standard and discussed 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 chapter 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-

eling approach. Instead they settled for the pragmatic, but less rigorous, loose (non-strict) metamodeling approach. This "adversely affected the integration of UML with other OMG modeling standards, such as the Meta Object Facility (MOF)" (Kobryn, 1999, p. 31).

As it is, The Object Management Group oversees standardization and it is through their processes that revisions of the UML are implemented. There are two mechanisms for standard revisions; RFPs and RTFs. The Request For Proposal (RFP) is the primary mechanism for updating specifications, while Revision Task Forces (RTF) are secondary. When a proposal is received, it is the RTF that examines and votes on the validity of it. The RTF is also able to recommend changes to the proposal in order to clarify areas that may be ambiguous. If the proposal is approved, then it becomes OMG adopted technology. If the proposal is not approved, then the RFP is reissued, with changes made to it to reflect the reasons for the last proposal failing.

Through the OMG system, a number of significant changes have been made in response to the some of the shortcomings identified in UML. Between UML 1.1 and UML 1.2, the specification was reformatted in order to better align with other OMG specifications. Typographical and grammatical errors were also targeted in this revision. Between UML 1.2 and UML 1.3, problems that had occurred during the alterations of the last revision were fixed, the activity graph notation was completed, and the standard elements were more formally organized. The revisions of the specification were all rather minor after that, up until UML 2.0 was released in 2005. The current UML Specification in use is UML 2.4.1, and unlike UML 1.x this specification is organized into four sections. These sections are called the Superstructure, the Infrastructure, the Object Constraint Language, and the UML Diagram Interchange. The Infrastructure "defines the foundational language constructs required" (OMG, 2011, p. 1). This is then balanced by the UML Superstructure, which "defines the user level constructs required" (OMG, 2011, p.1).

Studies have shown that the ongoing revision implementation has been successfully achieving the goal of standardization. As UML becomes more refined, it also becomes more universally accessible and accepted. A study found that 21% of Australian Computing Society members used UML frequently (Davies, 2006), further evidence of this growing standardization was demonstrated by Dobing and Parsons (2006) who noted that class diagrams were the most frequently utilized aspect of UML as reported by 73% of participants. Since these studies were conducted it has been demonstrated that practitioners have been successfully implementing the Unified Modeling Language more effectively and frequently, to such a point that it is now a part of many undergraduate university curricula in Information Technology fields. More recent studies have shown that the growth and uptake of UML has persisted over time (e.g. Dobing & Parsons, 2010; Budgen, Burn, Brereton, Kitchenham, & Pretorius, 2011).

The Future

The next specification for UML, UML 2.5, was due to be officially published on October the 4th, 2013. After failing to make a release by this date, an "In Progress" version of UML 2.5 was published in its stead. As such there is currently no set date on which this standard will be formally released.

According to the unofficial release specifications, there are a number of significant changes to be anticipated in this standard. Notably, the documentation format has once again been altered, so that "the UML Infrastructure is no longer part of the specification" (OMG, 2013, p. 3) – this means that instead of separate documents, it will once again be all contained within a single one. Another change that has occurred for this specification is that certain sections have been removed in order to help reduce the "bloating" that has been an issue of UML since its inception. As such, the notion of compliance levels has been eliminated as they were found to be ineffective. The repercussion of this is that now a tool either completely complies with UML notation or it does not – there is now no integrated compliance level. However, partial compliance can be achieved by "implementing a subset of its metamodel, notation, and semantics, in which case the vendor should declare which subset it implements" (OMG, 2013, p. 3).

Issues, Controversies, Problems

Despite years of revision that have successfully yielded incremental improvements to the specification, problems remain that need to be addressed. The standard elements are still rather "bloated" and they contain a level of inconsistency in both naming and organization. This level of complexity and the inconsistencies

introduced during revisions have been detrimental to the overall readability of the specification. There is a concern that the design of the notation is not sufficiently user-friendly, which would discourage potential users from adopting and using UML in favour of other simpler alternatives such as DOT graph description language, as described by Erickson and Siau (2013).

Another potential issue that needs to be addressed concerns the cyclic nature of specification revisions. In the process of updating UML to attempt to deal with the above issues and problems, excessive addition, removal and alteration of major concepts could affect the core structure of UML. The current method of revision leaves the core structure vulnerable. As previously stated, the UML focus of functionality was primarily for software engineering. As a result, software tools offer extensive support for UML when used with this focus in mind. However there is very little support in software tools for any other application, despite the language's goal of being a standardized representation of general-purpose models.

Solutions and Recommendations

The issue of "bloating" regarding the elements of UML is due to the inconsistent naming and organization throughout the standards. By phasing in more consistency to the various aspects of future specifications of the language, the volume of elements would be reduced and bloating would cease to be an issue. Reduction of elements and increase in consistency would also aid the language in terms of potential new users. Simplifying the language (and the specification documentation is relation to this) would increase the readability of it, and encourage more users to utilize UML instead of another modeling language. Also, core structure vulnerability can be corrected by the introduction of protocols within the specification revision procedure, to ensure that this remains unaltered. Support for non-software engineering projects is difficult to implement currently due to the fact that UML implementation seems to be "tool-based." If all of the various tools used for UML that are widely recognized started creating more support for the language in terms other than software engineering, then the specification would broaden to include this more as a result.

FUTURE RESEARCH DIRECTIONS

It has been shown that UML has been implemented within the field of software engineering increasingly over the years, moving from relatively low industry usage (e.g. Davies, 2006) to the present state where the growth in UML usage has led to an abundance of tools and software to better support the language. However, research into the Unified Modeling Language has been limited in recent years. There have been a few surveys conducted based on the use of UML in terms of software engineering and development, but very little in terms of its other applications as a general purpose modeling language. Current study seems to focus on the compliance of tools to UML, rather than the compliance of UML to its intended purpose. Future surveys of the adoption of UML (both within and outside of the field of software engineering) would be well served to include elements concerning the perceived effectiveness of the modeling language by users in real world situations. Another possible research direction may examine how the modeling language has affected development practices and utilization of techniques, and whether the overall project success rate has increased as a result of this.

CONCLUSION

The Unified Modeling Language may be the current industry standard, but it is still evolving and transitioning through constant revisions of the specification. These stages of revision are implemented to ensure that the UML remain effective and viable in the demanding and rapidly changing landscape of software engineering. This chapter examined this evolution in terms of three main periods described as The Past, The Present, and The Future. The Past section detailed the reason behind which standardization was needed, the history leading up to and including the development of UML. The initial reception from the user community and initial effectiveness were also discussed. The Present section then described the various changes between UML 1.0 and UML 2.4.1 and the reasons behind these changes and their ongoing effects. Finally, The Future section described the current state of UML, the expectations for the next specification of UML and also some open issues from the wider community which are yet to be

addressed. Some possible solutions and future research directions were also presented in light of these issues. In conclusion the Unified Modeling Language has proven itself to be an effective standard for communication and it will maintain its significant foothold in software engineering for the foreseeable future. However, the requirement for continual revisions to the specifications will also remain as the expected functionality and needs of UML practitioners will continue to chanve over time.

REFERENCES

Avison, D. E., & Fitzgerald, G. (2003). Where now for development methodologies? *Communications of the ACM*, *46*(1), 78–82. doi:10.1145/602421.602423

Booch, G. (1991). *Object-oriented design with application*. California: Benjamin-Cummings.

Budgen, D., Burn, A. J., Brereton, O. P., Kitchenham, B. A., & Pretorius, R. (2011). Empirical evidence about the UML: a systematic literature review. *Software, Practice & Experience*, *41*(4), 363–392. doi:10.1002/spe.1009

Cook, S. (2012). Looking Back at UML. *Software & Systems Modeling*, *11*(4), 471–480. doi:10.1007/s10270-012-0256-x

Davies, I., Green, P., Rosemann, M., Indulska, M., & Gallo, S. (2006). How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, *58*(3), 358–380. doi:10.1016/j.datak.2005.07.007

Dobing, B., & Parsons, J. (2006). How UML is used. *Communications of the ACM*, *49*(5), 109–113. doi:10.1145/1125944.1125949

Dobing, B., & Parsons, J. (2010). Dimensions of UML Diagram Use: Practitioner Survey and Research Agenda. In K. Siau, & J. Erickson (Eds.), *Principle Advancements in Database Management Technologies: New Applications and Frameworks* (pp. 271–290). Hershey: IGI Global.

Erickson, J., & Siau, K. (2013). *Unified Modeling Language: The teen years and growing pains. Human Interface and the Management of Information. Information and Interaction Design* (pp. 295–304). Berlin: Springer. doi:10.1007/978-3-642-39209-2_34

Fowler, M., & Scott, K. (2003). *UML Distilled*. Boston: Addison-Wesley.

International Organization for Standardization. (2005). ISO/IEC 19501:2005. Retrieved November 2, 2013, from http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32620

Jacobson, I. (1992). *Object-oriented software engineering: a use case driven approach*. Pearson Education.

Kobryn, C. (1999). UML 2001: A standardization odyssey. *Communications of the ACM*, *42*(10), 29–37. doi:10.1145/317665.317673

Lucas, F. J., Molina, F., & Taval, A. (2009). A systematic review of UML model consistency management. *Information and Software Technology*, *51*(12), 1631–1645. doi:10.1016/j.infsof.2009.04.009

Miles, R., & Hamilton, K. (2006). *Learning UML 2.0*. California: O'Reilly.

MOF Revision Task Force (1999). Meta Object Facility Specification v. 1.3. Document ad/99-06-05, Object Management Group.

Pilone, D., & Pitman, N. (2009). *UML 2.0 in a Nutshell*. California: O'Reilly Media.

Rumbaugh, J. R., Blaha, M. R., Lorensen, W., Eddy, F., & Premerlani, W. (1990). *Object-oriented modeling and design*. New Jersey: Prentice-Hall.

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M., & Wohed, P. (2006). On the suitability of UML 2.0 Activity Diagrams for Business Process Modeling. *Research and Practice in Information Technology*, *53*, 95–104.

OMG. (1999). *Policy and procedures of the OMG technical process. Document pp/99-05-01*. Object Management Group.

OMG. (2011). OMG *Unified Modeling Language (OMG UML), Infrastructure, v2.4.1*. Object Management Group. Retrieved November 10, 2013 from http://www.omg.org/spec/UML/2.4.1/Infrastructure

OMG. (2013). *OMG Unified Modeling Language (OMG UML), v2.5 FTF Beta1*. Object Management Group. Retrieved November 10, 2013 from http://www.omg.org/spec/UML/2.5

ADDITIONAL READING

Alhir, S. S. (2003). *Learning UML*. California: O'Reilly.

Álvarez, A. T., & Alemán, J. L. F. (2000). Formally modeling UML and its evolution: A holistic approach. In S. F. Smith (Ed.), *Formal methods for open object-based distributed systems IV* (pp. 183–206). US: Springer. doi:10.1007/978-0-387-35520-7_9

Bennett, S., McRobb, S., & Farmer, R. (2006). *Object-oriented systems analysis and design using UML* (Vol. 2). Berkshire, UK: McGraw-Hill.

Eriksson, H.-E., & Penker, M. (1997). *UML toolkit*. John Wiley & Sons, Inc.

Eriksson, H.-E., & Penker, M. (2000). *Business modeling with UML*. Chichester: Wiley.

Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Languange*. Boston: Addison-Wesley Professional.

Kobryn, C. (1999). UML 2001: a standardization odyssey. *Communications of the ACM*, *42*(10), 29–37. doi:10.1145/317665.317673

Miles, R., & Hamilton, K. (2006). *Learning UML 2.0*. California: O'Reilly.

Object Management Group. (2007). *Unified Modeling Language (OMG UML)*. Superstructure.

Pooley, R., & Stevens, P. (1999). *Using UML: Software Engineering with Objects and Rules*. Boston: Addison-Wesley Longman.

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The Unified Modeling Language reference manual*. Boston: Addison-Wesley Longman.

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2005). *Object-oriented Analysis and Design: With the Unified Process*. Boston: Thomson Course Technology.

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2011). *Systems analysis and design in a changing world*. Boston: Cengage Learning.

KEY TERMS AND DEFINITIONS

Model: A conceptual diagram used to represent a system.

Object Management Group: An organization created with the goal to determine a standard method of communication between distributed objects.

OOPSLA: "Object-Oriented Programming, Systems, Languages and Applications" – an annual research conference run by the Association for Computing Machinery.

Software Engineering: The application of systematic methods and approaches for the development and maintenance of software artifacts.

Specification: The set of requirements that must be satisfied in order for any model to comply with the current standards of UML.

Unified Modeling Language: A form of notation developed with the core goal of creating a standardized representation of general-purpose models, with the focus of functionality primarily being for software engineering.