2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion

Towards Promoting Design and UML Modeling Practices in the Open Source Community

Abdullah Aldaeej
State University of New York At Albany
1400 Washington Ave
Albany, NY 12222
aaldaeej@albany.edu

Omar Badreddin
Northern Arizona University
S San Francisco St
Flagstaff, AZ 86001
omar.badreldin@nau.edu

ABSTRACT

Despite the emergence of UML as the defacto modeling and design tool for software engineering, its adoption remains dismal. Software development, particularly in the open source community, remains code-centric. Adoption of UML in open source projects represents a significant lost opportunity. In this paper, we present an approach to encourage upfront design practices and the adoption of UML modeling in open source projects. In particular, we demonstrate the approach for small contributions and bug fixes. The approach relies on integrating UML-level abstractions into the code. This integration means that open source developers can continue to use their familiar text-based tools to manage the source code and contributions, while at the same time benefit from UML added value of abstractions and comprehension. Other benefits of this approach include broadening the boundaries of bug fix contribution by including modelers and end-users, and incrementally add UML model diagrams into open source project's documentation.

CCS Concepts

• Information Systems → Open source software • Software and its engineering → Model-driven software engineering • Software and its engineering → Unified Modeling Language (UML)

Keywords

Umple; Model Oriented Programming; Open Source Projects; UML; Program Comprehension; Bug Fixing; Software Design; Reserve Engineering; Forward Engineering; Code Generation.

1. INTRODUCTION

The adoption of modeling tools is absent in open source communities [2, 9]. Based on the survey in[3], the reasons behind this absent include the complexity of the modeling tools, and the difficulty to overcome the prevalent code culture. In addition, Open Source Software (OSS) graphic modeling tools are improving slower than other software development technologies, and lack of visual tools that support OSS projects[6]. Besides the low adaption of modeling, Open Source (OS) developers and contributors can only commit code segments. Once a bug issue is established, the contribution is limited to code developers to improve the code.

ICSE '16 Companion, May 14 - 22, 2016, Austin, TX,USA

ISBN 978-1-4503-4205-6/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2889160.2892649

Umple, an open sourced model oriented programming language, was developed to help open source developers adopt modeling by 1) continue to use textual repositories such as SVN and GIT, 2) provide an environment familiar to code-centric developers, and 3) integrate modeling abstractions with code[2]. In this paper, we present an approach of using Umple tool in OSS projects as a way to incorporate UML modeling during the bug fix process. This approach would expand and diversify the contribution of bug fixing, including modelers and end users. The remaining of this paper is organized as follows. In section 2, we present related literature in Umple and modeling practices in OS community. In section 3, we explain our approach of adopting Umple in OSS projects. In section 4, we discuss the limitation of the approach. In section 5, we conclude the paper.

2. BACKGROUND AND LITERATURE REVIEW

In this section, we give an overview of Umple and the process to reverse engineer existing source code into the model oriented language Umple. We also provide an overview of modeling practices in OSS community.

2.1 The Umple Model Oriented Language

Umple [10] is an open source modeling and programming language that aims to enable model-oriented programming. It adds UML associations, attributes, and state machines to object-oriented programming languages such as Java, C++, PHP. Umple has been developed by a team in the University of Ottawa to facilitate modeling adoption in software engineering projects especially OSS projects. It is written in itself by manually refactoring the original java version into Umple; and has been published as open source since 2010[2].

Umple supports incremental and rapid development environment [4], by combining high level system abstraction and low level specification in the same development artifact. It is also used for teaching UML modeling in classrooms [8]. Umple has also a web-based environment [11] that allows users to model either textually or visually, and where the changes are automatically synchronized.

2.2 Umplification

Umplification is a process of converting base language program into Umple program through a set of re-factorings [7]. This process resembles the performance of any reverse engineering tool, but the unique difference is that the resulted model is not a separated artifact. The final result of this process is a model that semantically integrated with the code, significantly facilitating round-trip engineering. The purpose of the Umplification process is to: 1) allow a better understanding of the program, 2) reduce the volume of code, and 3) facilitate the switch to model-driven development [5].

The Umplification process takes three main transformation steps [5, 7]. The following is a summary of these transformations (adopted from [5]):

- **Transformation 0:** renaming the source files (e.g. Java, C++) as Umple file with .ump extension.
- **Transformation 1:** changing the sub-classing notation into the Umple 'isA' notation; packaging notation into Umple 'namespace' notation; dependency notation such as 'import' into Umple 'depend' notation.
- **Transformation 2:** analyzing and converting instance variables to reflect Umple attribute, association, or state machine.

The Umplification process can be semi-automated by using a rule-driven tool called "Umplificator" [5]. This tool is a language interpreter and static analyzer that parses the base language code and processes it into an internal representation. It relies on initial parsing tools such as Java Development Tool (JDT) for Java to extract the model from the base language code. The Umplificator tool then applies a predetermine set of refactoring rules to transform the language model into Umple model. The tool integrates mapping rules for the transformation 1, and some of the transformation 2.

2.3 Modeling Practices in OSS Community

The Modeling adoption in OSS projects remains very low [2, 9]. OS developers favor to deal with code rather than model for many reasons. According to the study in [3], OS developers resist to adopt modeling because of the high complexity of modeling tools, and the difficulty associated with changing the code culture in OSS community. OS developers also claim that models would not generate quality code that satisfies their needs [3]. Another study reveals that OSS modeling tools are improving slower than the other software development technologies, which further slows the adoption of modeling in OSS projects [6].

Umple tool can facilitate the adoption of modeling in OSS projects since it includes Java code, UML association, and UML state machine in the same textual artifact [2]. Umple would help the OSS community by shrinking the amount of code [2]; and incorporates visual diagrams into the project documentation; having these visual diagrams helps attracting new OS contributors, and benefiting the software maintenance process [6]. Therefore, we believe that adopting Umple can have significant benefit to the bug fixing process in OSS projects.

3. THE PROPOSED APPROACH

We propose an approach of using Umple in OSS projects during the bug fix process. We specify the modeling practice at this process for three reasons: 1) bug process is an integral part of OSS projects activities, 2) promote diversity in bug fixing contributors, including modelers and end-users, and 3) incrementally incorporate modeling diagrams in OSS projects' documentation.

Umple will be used as a tool to perform the modeling in this approach [1]. We select Umple because it is well suited for modeling in such OSS environment [2]. In this approach, the modeling practice begins after a bug request is initiated and validated. OSS project maintainers would refactor (*umplify*) the segment of code that corresponds to the bug. Then, automatically generate UML model (class & state machine diagrams), and save the result as an Umple file. Both the original bug code and the Umple file are to be stored in the bug report.

This approach would enable two types of contributions. Coders who are comfortable with code more than models can directly access the original source code and disregard the model. Then, they can submit their code solution, as in the traditional bug fix process. Modelers who are more familiar with model than code can access Umple file to start with the model. Then, automatically generate code from the modified model as a bug solution. The project maintainers will receive the bug solution as either Umple file or patch code. If the contribution comes as an Umple file, then both model and code are committed. However, if the contribution comes as a patch code (from coders), the master copy of Umple file needs to be updated to reflect the new changes before committing model and code. The figure 1 below illustrates our proposed approach.

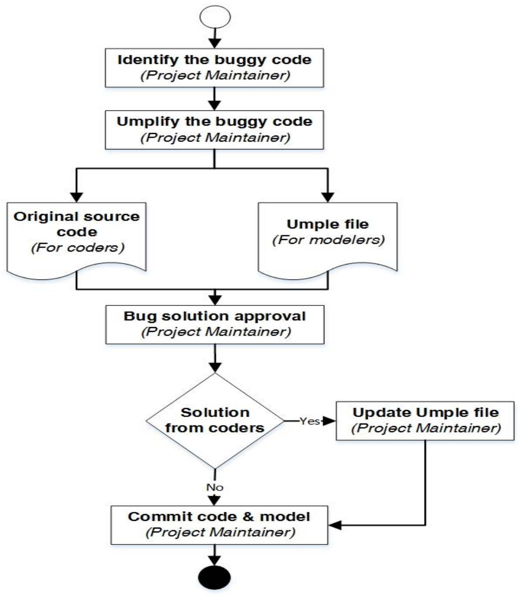

Figure 1: Overview of the proposed approach

4. LIMITATION OF THE APPROACH

The approach suffers from some issues that might affect its performance. First, it requires that OSS projects use object-oriented programming language for their source code in order to comply with UML modeling notation. In addition, the approach is applicable for OSS projects that have a maintainer (committer) team who validate the bug issues and identify the bug code beforehand. Lastly, the approach needs to be tested empirically in real OSS cases.

5. CONCLUSION

The majority of OSS projects remain code dependent to build or fix source code. We present an approach to facilitate UML modeling practices in OSS projects utilizing Umple tool. The approach incorporate modeling practice during the bug process in order to incrementally add modeling abstraction to the source code, and broaden the contribution circle to include modelers and end-users. For future work, we plan to conduct empirical studies by integrating this approach into an OSS project, and validating the efficiency of the bug process in term of quantity (number of contributors) and the quality of the bug fix.

6. REFERENCES

[1] Badreddin, O., Forward, A., and Lethbridge, T. C. 2012. Model oriented programming: an empirical study of comprehension. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*. IBM Corp.

[2] Badreddin, O., Lethbridge, T. C., and Elassar, M. 2013. Modeling Practices in Open Source Software. In *Open Source Software: Quality Verification*. Springer Berlin Heidelberg,127-139.

[3] Forward, A., Badreddin, O., and Lethbridge, T. C. 2010. Perceptions of software modeling: a survey of software practitioners. In *5th workshop from code centric to model centric: evaluating the effectiveness of MDD (C2M: EEMDD)*.

[4] Forward, A., Badreddin, O., and Lethbridge, T. C. 2010. Umple: Towards combining model driven with prototype driven system development. In *the 21st IEEE International Symposium on Rapid System Prototyping (RSP)*. IEEE, Fairfax, VA, 1-7. DOI=http://doi: 10.1109/RSP.2010.5656338.

[5] Garzón, M. 2015. *Reverse Engineering Object-Oriented Systems into Umple: An Incremental and Rule-Based Approach*. Ph.D. Dissertation. University Of Ottawa. Ottawa, Ontario.

[6] Kim, W., Chung, S., and Endicott-Popovsky, B. 2014. Software architecture model driven reverse engineering approach to open source software development. In *Proceedings of the 3rd annual conference on Research in information technology*. ACM, New York, NY, 9-14. DOI=http://doi:10.1145/2656434.2656440.

[7] Lethbridge, T. C., Forward, A., and Badreddin, O. 2010. Umplification: Refactoring to incrementally add abstraction to a program. In *Proceeding of the 17th Working Conference on Reverse Engineering (WCRE)*. IEEE, Beverly, MA, 220-224. DOI=http://doi:10.1109/WCRE.2010.32.

[8] Lethbridge, T. C., Mussbacher, G., Forward, A., and Badreddin, O. 2011. Teaching UML using umple: Applying model-oriented programming in the classroom. In *Proceeding of the 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, Honolulu, HI,421-428. DOI=http://doi:10.1109/CSEET.2011.5876118.

[9] Robbins, J. 2005. Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools. In *Perspectives on Free and Open Source Software*. MIT Press, 245 – 264.

[10] Umple: Merging Modeling with Programming. Retrieved October 15, 2015 from http://cruise.eecs.uottawa.ca/umple

[11] Umple Online. Retrieved October 15, 2015 from www.Try.Umple.org