Empirical Evaluation of Research Prototypes at Variable Stages of Maturity

Omar Badreddin
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
obadr024@uottawa.ca

Abstract— **Empirical evaluation of research tools is growing especially in the field of software engineering. A number of research techniques have been proposed and used in evaluating research prototypes. We take the view that evaluation of software engineering tools is best achieved in industrial settings, with real life artifacts and tasks, and with professional software engineers. However, the feasibility of such evaluation is limited for many reasons. Some challenges are related to the prototypes under study, others are related to the industrial environments where the need to meet business requirements take precedence on experimenting with new tools and techniques.**

In this paper, we summarize our experiences in evaluating a research prototype tool using a grounded theory study, a questionnaire, and a controlled experiment. We discuss the challenges that hindered our industrial evaluation and share ideas on how to overcome these challenges. We propose an action research study where the research tool is used by a small number of experienced professionals in an industrial project.

Keywords- UML, Modeling, Umple, Empirical Studies, Grounded theory, Controlled experiment, Action Research.

I. INTRODUCTION

Evidence based software engineering is a relatively young research domain. Social and life sciences rely more extensively on empirical investigations and evaluations and have developed mature guidelines and procedures to conduct such studies. In fact, a number of empirical methods, such as grounded theory [1], have emerged from the social sciences domains.

In software engineering, and based on our experiences, we find that certain research methods tend to be more successful for certain evaluation tasks. For example, we find that controlled experiments are well suited for the evaluation of research prototypes. This is mainly due to the following:

- In controlled experiments, unintended variability can be controlled. For example, problem complexity and technical expertise of participants can be controlled to minimize or eliminate their interference with experimental variables.
- Tool limitations can be concealed. This can be achieved either by limiting the scope of the experiment, by using paper prototypes, or by simulating other features of the tool under study.

Controlled experiments, however, suffer from external validity. Even if the experiment was conducted with professional participants, it is hard to confidently make claims about how the tool or approach will scale up to larger and more complex problems.

Fortunately, there are other research methods that can be utilized to study software engineering practices and tools in real life. The literature is rich with successful implementations of such methods. Examples of such methods include shadowing, action research, grounded theory studies, and interviews.

Many methods, having emerged from the social sciences, tend to be more successful when the subject of the study is a phenomenon or a behavior, and not a research prototype tool. When the subject of the study is the research tool itself, it becomes more challenging to conduct an empirical study. The majority of the research tools are functionally incomplete and cannot be naturally adopted and evaluated in a way similar to studying a phenomenon or a behavior.

In this paper, we summarize our experiences in conducting studies with a research prototype tool at varying stages of maturity and over a number of years. At early stages, we have conducted a grounded theory study [1], interviews and questionnaires with early adopters. Attempts to evaluate the tool in industry using a case study method did not succeed at this stage. When the tool became more mature, we have successfully conducted a controlled experiment with users that included both student and professional participants [2]. The controlled experiment revealed valuable results and insights into the nature of the ideas embodied in the tool. However, the main criticism of the work was related to the experiment artifacts and tasks being too simple. Technical limitations in the tool were one of the main reasons why it was not feasible to utilize larger and more complex artifacts. Since then, the tool maturity has improved, and the number of early adopters has increased. We are now in the process of designing an action research study [3].

This paper is organized as follows. First, we give a brief background on the tool under evaluation. Then, we present a summary of our previous work that includes a grounded theory study, a questionnaire study, and a controlled experiment. We then present the outlines of our planned industrial case study. In this section, we discuss the industrial project selected for the study and how we plan to go about designing and executing the study. Finally, we conclude by

summarizing some of the common challenges research prototype tools face when conducting empirical and industrial evaluations.

II. Background: The Umple Technology

A key philosophy behind Umple is that modeling and coding are complementary. In other words, both model and code artifacts add value to the development of a system. It is true that modeling is typically visual and coding is typically textual, but it does not need to be this way exclusively. Umple enables users to model textually and code visually. The best way to demonstrate this is by an example.

```
1   class Person { }
2
3   class Student {
4     isA Person;
5     Integer stNum;
6     status {
7       Applied {
8         withdraw -> Withdrawn;
9         enroll [!hold] -> Enrolled;
10      }
11      Enrolled {
12       withdraw -> Withdrawn;
13       graduate-> Graduated;
14      }
15      Graduated {}
16      Withdrawn {}
17    }
18    * -- 0..1 Supervisor;
19  }
20
21  class Supervisor {
22    isA Person; }
```

Listing 1: Sample Umple code/model [2]

Consider the sample system in Listing 1. The lines of code look all familiar to a Java developer, except for lines 4, 6 to 18, and 22. These lines represent modeling elements embedded within the Java code.

Line 18 indicates that class Student has a many-to-optional relationship with class Supervisor. Similarly, lines 4 and 22 indicate that the classes Student and Supervisor, respectively, inherit from class Person. These modeling elements are typically represented visually in a UML class diagram, such as the one in Figure 1.

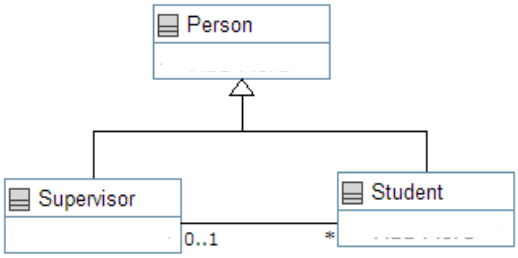

Figure 1: A class diagram in UML

Lines 6 to 17 describe the behavior of the Student class with a state machine that resembles very closely one modeled in UML. The equivalent UML state machine model for these lines of Umple code is depicted in Figure 2. When a student object is created, it starts at the *Applied* state. The state machine then specifies different states it goes to in response to events. For example, when the event *withdraw* occurs, the student object goes into state *Withdrawn*.

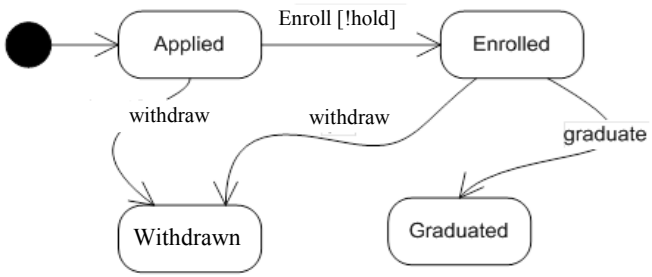

Figure 2: State machine model

The purpose of this example is to demonstrate the concept of embedding modeling elements within the code. Not mentioned here are the interfaces generated by these modeling abstractions, as well as modeling of other UML abstractions. Umple is an open source project [4]. For more information on Umple, the reader is encouraged to refer to the already published resources [5, 6].

III. Previous Empirical Studies

In this section, we summarize three empirical studies of Umple. The first is a grounded theory study of Umple users, the second is a questionnaire of Umple's early adopters, and the third is a controlled experiment with student and professional participants.

A. Grounded Theory Study of Umple Users

Grounded theory is useful as an exploratory study with little or no need for assumptions or hypotheses. This method, we find, is particularly useful when the research prototype tool is at early stages of development. In our case, it was a good first step in exploring users' perceptions and in collecting feedback. We utilized findings from this study to guide the experimental development.

The study entailed a number of interviews with participants who had variable experiences with Umple. We asked open ended questions to encourage exploration of users' perception and feedback. The interview questions are related to experiences using Umple, major challenges, and modeling experiences.

The results of this study indicate that users perceive textual modeling to be of less complexity than the equivalent Java code for a subset of tasks. Users also shared their view on the generated code, the textual modeling paradigm, and other aspects of the platform such as error reporting and syntax highlighting. The results of this study were used in two ways; first, to determine a few high level

objectives against which design decisions were made. For example, one objective is to reduce the number of keywords in the language on the assumption that fewer keywords results in a language that is less complex and easier to learn. Second, the results were used to incrementally refine the implementation of the language.

Finding enough participants who had reasonable experiences with Umple proved to be challenging. The results of this study were useful for us, the researchers behind developing Umple. However, these results were of little scientific value.

B. *Questionnaire study*

Interviewing early adopters was useful in discovering their perceptions and experiences. In this study, we attempted to collect more quantitative data, and to focus on specific aspects and concepts embedded in the research tool. A questionnaire study was our choice to achieve these two goals. Anonymous participants were conveniently selected from the pool of early adopters. The questionnaire put very little emphasis on the research tool specifically, but rather addressed the tool approach and method. This enabled us to widen the scope of potential participants. We were able to involve participants who had little to no experience with our research tool. The results of the questionnaire complemented our grounded theory study. It was by then evident that we needed to experiment with the research tool itself. Our next step was a controlled experiment.

C. *Controlled Experiment*

This experiment was the most successful study of our research tool so far. We were able to control variables that we did not want to influence our findings. We were also able to limit the scope of the experiment and to compare our research tool with other comparable modeling and development environments. In this section, we give a brief overview of the experiment. Experiment specifications and results are published [2].

The experiment focused on the following main hypothesis:

H1: A system written in Umple is more comprehensible than an equivalent Java implementation of the system. In other words, participants take on average less time to respond to questions when presented with an Umple version of a system as opposed to a Java version.

The experiment consisted of three rounds of comprehension questions that measure the effectiveness of the notation. Each round contained 12 questions. Participants usually provided responses within 30 seconds of posing the question. Some questions addressed the concept of associations as present in a class diagram or a textual notation. Other questions addressed comprehension of a state machine.

The results of the experiment indicate that Umple's notation was at least as good as UML visual notation, and is statistically better than Java. However, like with many controlled experiments, external validity was a major threat. The main concerns of this work were the overly simple experiment artifacts, questions, and tasks.

The challenges of experimenting with more complex or real life artifacts or tasks are the technical limitations in the research tool under study. For example, Umple has limited support for debugging and error messages. If participants use Umple to build satisfactory complex systems, their efficiency and performance will inevitably be affected by the limited debugging facilities provided by the Umple platform.

This limitation has motivated us to design an action research study where Umple is used by a few experts who are well familiar with the tool and its limitations and who can work with the existing platform relatively unchallenged by its restrictions. The next section gives the broad outlines of this planned study.

IV. Industrial Action Research Study

Action research is an empirical method whereby the researcher becomes actively involved with professionals in their daily tasks. The researcher can therefore have firsthand experience in how the research tool can be deployed and used by professionals.

We plan to use Umple in an industrial project to build some components of a hospital management system. In this section, we outline the properties of the project that we hope will facilitate the action research study. We also give broad outlines of how we plan to assess the research tool. However, the specifics of the study are still being determined. We plan to enhance and refine the design of the study along the way.

A. *The host project*

A patient and hospital management system is under development by a team of professionals and researchers. It is a multi-year project with a relatively large number of components. We plan to deploy Umple to build an event-driven sub-component that is concerned with constructing complex state machine models of patients flow. By collecting information about patient location, events, wait time, and other relative context information, the system must identify the patient state, calculate wait time, and assess patient flow efficiency. This component is driven by a complex set of events that controls the state machine transitions. The transition will involve, among other things, complex guard conditions.

We have determined that Umple is a suitable candidate for this project for the following reasons.

- A multi-year project gives the researchers enough time to deploy Umple and experiment with the development of the required component. If a key feature is missing, or if blocking bugs emerge, then the team will have time to provide the required fix.
- Umple's support for state machine is sophisticated and includes composite state machines and complex

guard conditions. Both are a requirement for the development of the targeted component.

- Umple supports the rapid development of prototype running systems [7]. We anticipate that this feature will enable the team to quickly assess the feasibility of the approach.

B. *The Action Research Study*

Two software engineers will be responsible for developing the sub-component. Specifications for the interface of the component will be provided to the team. The team will build the component using Umple, and generate a complete system in Java, and integrate with the complete patient flow management system. The software engineers will have access to the Umple open source project. Emerging bugs will be tracked and addressed as resources permit.

C. *Expected results*

We hope to achieve the following goals. First, we aim to evaluate Umple's approach in an industrial system, discover any limitations, and enhance the overall quality of the tool. Second, and more importantly, we aim to study how professional engineers perceive and utilize textual UML modeling. Third, we expect to broaden the number of the users and possibly encourage adoption in the industry.

V. DISCUSSION AND CONCLUSION

Empirical evaluation of research prototypes is very challenging, particularly because experimental tools lack the required maturity for effective evaluation in real life or synthetic tasks. Our experiences indicate that different evaluation methods should be used at different stages of tool maturity. At very early stages, a grounded theory study or interviews of early adopters seem to work very well because they put less emphasis on the tool and more emphasis on the users' perceptions and experiences. This approach can have very little involvement of the tool under study. It becomes more feasible to conduct controlled experiments when the tool under study is more mature. At this stage, the tool must support some core use cases with high quality to facilitate the execution of the experiment. When the research tool is able to support a larger number of use cases with reliable and user friendly interfaces, then a more rigorous evaluation becomes possible. We anticipate that involving the tool in an action research in the industry becomes possible at this stage.

It is imperative for the software engineering research community to acknowledge both the need and the challenge in experimenting with emerging prototype tools. The community must accept limited experiments and limited results for tools that are at early stages of maturity. Using artificially small artifacts for evaluation, despite posing numerous threats of validity, is valuable in the evaluation of emerging prototype tools. We believe that numerous researchers face challenges similar to the ones that we faced. Sharing experiences of failure is as important, if not more, as sharing success stories.

REFERENCES

[1] Hansen, B. H. and Kautz, K. "Grounded Theory Applied-Studying Information Systems Development Methodologies in Practice," in *Proceedings of 38th Annual Hawaiian International Conference on Systems Sciences,* 2005. pp. 264-264.

[2] Badreddin, O., Forward, A. and Lethbridge, T. C. "Model Oriented Programming: An Empirical Study of Comprehension". 2012. *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research,* pp. 73-86.

[3] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., et al. "Preliminary Guidelines for Empirical Research in Software Engineering". 2002. *IEEE Trans.Software Eng.,* IEEE Computer Society. pp. 721-734.

[4] Lethbridge, T. C., Forward, A. and Badreddin, O. "Umple Google Code Project". 2012. Available: code.umple.org

[5] Lethbridge T.C., Forward, A. and Badreddin, O. " Umple Language Online.", accessed 2012, http://try.umple.org.

[6] Badreddin, O. "Umple: A Model-Oriented Programming Language," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2,* 2010. pp. 337-338.

[7] Forward, A., Badreddin, O., Lethbridge, T. C. and Solano, J. "Model‐driven Rapid Prototyping with Umple". 2011. *Software: Practice and Experience,* Wiley Online Library.