Improving Program Comprehension by Enhancing Program Constructs: An Analysis of the Umple language

Andrew Forward, Timothy C. Lethbridge, and Dusan Brestovansky
School of Information Technology and Engineering (SITE), University of Ottawa, Canada
{aforward, tcl}@site.uottawa.ca, dbres042@gmail.com

Abstract

Umple is a set of extensions to object-oriented languages that provides a concrete syntax for UML abstractions like associations. We argue that Umple will help increase program comprehension by allowing developers to describe a system at a more abstract level, and by reducing the volume of code.

1. Introduction and motivation

Umple is a programming language technology whose objective is to simplify software by incorporating modeling abstractions like associations.

A first motivation for our work is to make programs easier to understand and write. Mayrhauser [1] states that to understand a program, you need a mental representation of both its low-level technical artifacts, as well as how these artifacts relate to the application domain. Umple directly incorporates UML associations, which map directly to the application domain. Using Umple, program comprehension tools (such as those in [2]) would require fewer inferences to match domain concepts with relevant source code.

A second motivation is that most developers remain code-centric, hence visual modeling tools are not as widely adopted as might be desired. Developers often draw diagrams when explaining concepts in design meetings, and include some of them in documents, but relatively few use models to actually *drive* development. This is particularly true in the agile and open source communities [3].

In a survey we conducted [4], 68% of respondents felt it is bad or terrible that models become out of date or inconsistent with the code; 52% complained about incompatibility among modeling tools, and 39% complained about modeling tools being too heavyweight.

On the other hand the respondents also had complaints about code-centric approaches: Two thirds complained about difficulties understanding the design or behavior of the system based on code, and over half complained about code being difficult to change in general, as compared to models.

We conclude from our analysis that the reasons for lack of adoption of modeling seem to be: a) code generation is often inadequate; b) modeling tools are too difficult to use; c) a culture of coding prevails and is hard to overcome; d) there is a lack of understanding of modeling; and e) the code-centric approach works well enough, such that the return on investment of adopting modeling is marginal, yet the risks high.

Umple addresses all these points. Point a) is addressed since Umple is a programming language, so generated code does not need to be examined. Umple addresses point b) by allowing models to be created using a text editor. A syntax-directed editor can hence be used to help produce error-free models. The Umple tools [5] enable rendering Umple code as UML diagrams directly and instantly. Umple addresses point c) because it does not try to go against the prevalent coding culture. It allows you to keep 'coding' while you are modeling. The user therefore has the 'best of both worlds' – programming and modeling combined.

Umple addresses point d) by only introducing the simplest modeling concepts: associations (with multiplicity) and attributes. Finally, Umple addresses point e) by providing a path to adoption of modeling that doesn't require a major investment.

A final motivation for our work is that there is much repetitive code in object-oriented programs. Hence we wish to incorporate abstractions to promote understandability and reduce code volume.

Long before the advent of UML it was a commonplace object-oriented programming idiom for two classes to contain instance variables that reference the opposite class. The programming challenges included maintaining referential integrity, and deciding which class would take the responsibility for adding and deleting the references (links) between objects.

With the emergence of object-oriented modeling languages such as UML, these between-class

references were modeled as *associations*. Detailed semantics can be found in the UML specification [6].

Ultimately, associations have to be rendered into programming language code. Developing bug-free code to do this is not easy, yet the code ends up being similar from association to association. It is called *boilerplate* code because it is standard in nature, yet needs to be replicated in many parts of a system. With Umple, this boilerplate code for associations is completely eliminated. Instead one declares associations and lets the compiler take care of the rest.

2. Overview and Example of Umple

An Umple program contains algorithmic methods that look more or less identical to their Java or PHP counterparts. The key differences lie in a) how classes are declared, b) the absence of explicit constructors, and c) the replacement of instance variables and many related methods by *attributes* and *associations*.

The following example shows how one would declare attributes and associations in the first steps on modeling a system using Umple.

```
class Student {}

class CourseSection {}

class Registration {
  String grade;
  * -- 1 Student;
  * -- 1 CourseSection;
}
```

As this shows, the basic declaration of a class follows the syntactic style of Java. The three lines in class Registration declare an attribute and two associations respectively. The class diagram to reflect the Umple code appears in Figure 1.

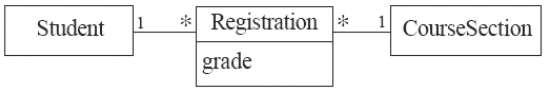

Figure 1: UML class diagram for part of the student registration system (from [7])

Please refer to [8] for full details about Umple. The UmpleOnline web application [5] allows you to *try out* Umple without having to install any software on your local computer. Simply type in Umple code and generate UML diagrams as well as the Java (or PHP) implementation. You can also edit the UML diagrams and see the change reflected in Umple code.

We have written several systems in Umple, including version 1.3 of Umple itself. In this, Umple code for associations and attributes replaced about 43% of the original Java code. Typically the amount of Umple code needed for associations and attributes is 10% of the amount of Java or PHP code needed. Other examples of systems we have developed are an airline reservation system and a simple elevator simulation (to test Umple in the real time domain).

Umple currently generates the core of a system with an API that can be called by other layers. It does not generate the user interface. One of Umple's strengths is that it helps enforce a strongly layered architecture.

3. Conclusions

Umple is a programming language technology, applied to both Java and PHP, that incorporates UML concepts such as associations and attributes. It is also an environment to enable creation of UML-like models textually.

We created Umple to respond to three needs. The first is to make program creation and comprehension easier; the second is to combat the resistance to modeling prevalent in industry, and the third is the desire to eliminate boilerplate code.

We have shown that full-fledged systems can be built using Umple. Indeed we have used Umple to develop itself.

We intend to enhance Umple with capabilities for user interface generation, state machines and a pattern-based plug-in mechanism.

References

[1] A. von Mayrhauser and A. Vans, "Industrial experience with an integrated code comprehension model" IEE Software, Sept 1995, pp. 171-182.

[2] Clayton, R., Rugaber, S., Taylor, L. and Wills, L. (1997) "A case study of domain-based program understanding," *IWPC*, pp. 102-110.

[3] G. Hertel, S. Niedner and S. Herrmann, "Motivation of Software Developers in Open Source Projects: an Internet-Based Survey of Contributors to the Linux Kernel", Research Policy 32, 2003, pp. 1159-1177.

[4] A. Forward, "Perceptions of Software Modeling: A Survey of Software Practitioners", 2007, http://www.site.uottawa.ca/~tcl/gradtheses/aforwardphd/

[5] "UmpleOnline," http://cruise.site.uottawa.ca/umpleonline/

[6] "Object Management Group, Unified Modeling Language (UML), version 2.1.2," accessed March 2009, http://www.omg.org/technology/documents/formal/uml.htm

[7] T.C. Lethbridge, and R. Laganière, *Object-Oriented Software Engineering: Practical Software Development using UML and Java*, 2nd Ed., 2004, McGraw Hill.

[8] "Umple Language," http://cruise.site.uottawa.ca/umple/