Umple: An Open-Source Tool for Easy-To-Use Modeling, Analysis, and Code Generation

Timothy C. Lethbridge

School of Electrical Engineering and Computer Science
University of Ottawa, Canada K1N 6N5
tcl@eecs.uottawa.ca

Abstract. We demonstrate the Umple technology, which allows software developers to blend abstract models, including class-, state- and composite structure diagrams textually into their Java, C++ or PhP code. Umple is targeted at developers who prefer textual programming but also want additional abstractions in order to simplify their software and improve its quality. Umple development has involved over 60 people, mostly at Canadian and US universities, and is used to develop itself. Several systems have been umplified – converted into Umple – thus raising their abstraction and reducing code volume. The accompanying video can be found at http://youtu.be/xD-zTpB_zyQ .

Keywords: Code generation, Textual Modeling, Umple, UML, State Machines

1 Introduction

Umple is a multi-faceted technology allowing users to integrate modeling into software development straightforwardly. It supports modeling using class diagrams, state machines and composite structure diagrams, and provides a textual syntax for these that can be blended into any C-family language such as Java or C++. The resulting system can consist completely of modeling abstractions, completely of base programming language code, or a blend of either. The Umple textual form is the 'master' code for the system. Umple therefore renders the distinction between model and code somewhat moot.

Umple can display and update model diagrams as text is edited, and allows changes to diagrams to automatically change the Umple text. This is accomplished in near-real-time using UmpleOnline [1]. The developer can hence work productively, whether they prefer text or diagrams.

Umple supports a rich feature set, all documented with examples in its user manual [2], and all generating fully-operational code in Java and C++. Features include:

- UML **associations** with capabilities such as referential integrity, sorting, and enforcement of multiplicity constraints [3].
- **State machines** with unlimited nesting, concurrent activities, and a choice of implementation semantics such as having a separate thread for queuing events [4].

- **Traits** to support inclusion of model or code fragments in different contexts, or to overcome lack of multiple inheritance.
- **Active objects** and **ports** for communicating among concurrent objects (including support of parts of Autosar [5]).
- **Constraints** for invariants, state transition guards, method preconditions and ports.
- Built-in **patterns** such as singleton and immutable, with idioms for other patterns such as delegation.
- **Aspect-oriented** code injection to allow tailoring of the generated code.
- **Templates** to allow construction of string output for language generation.
- **Trace-directives** to allow dynamic analysis at the model level [6].

Umple supports mixins to allow the system to be structured in several ways. These include separating model abstractions from methods of classes, or dividing up the system in a feature-oriented manner. With mixins, multiple definitions of a given model element (e.g. a class) found separately in the Umple source files, are merged.

Umple can generate C++. Java, PhP, Ruby, SQL, metrics, documentation and various model-interchange formats such as ECore XMI, USE , TextUML and YUML. Particular focus is being placed on its ability to generate real-time systems.

Umple is under active development. Upcoming features include formal method generation, incorporation of Use Cases, requirements , and product-line capabilities. Umple has been designed to be extensible; new code generators and modeling concepts can be added easily – a process that has been going on for the last 7 years.

2 Envisioned Users

Umple is intended for general-purpose development, so anybody currently developing in one of Umple's primary supported languages can use it to enhance productivity. Anyone who wants to model using UML class diagrams, state diagrams or composite structure diagrams can also use it purely for that purpose, even if they don't intend to generate code. However, Umple is particularly targeted at the following groups:

- **Open source developers and small in-house developers**: For these communities, code is king. They may use a little UML on whiteboards, but they don't generate code due to awkward or expensive tools, or poor quality of the code generated by many tools.
- **University professors and students**: Umple is designed to be as easy to use as possible to facilitate teaching and learning, as discussed in the next section.
- **Developers who want the flexibility and the minimum of dependency**: There are several ways of structuring an Umple system, and it can be managed with many tools: Umple supports command-line, Eclipse-based and web-based development. Umple generated code doesn't require linking with third-party libraries. Although Eclipse's EMF is powerful, we avoided it to preclude dependency on Eclipse.
- **Developers who want generated code that is readable (and inspectable), but need to avoid modifying it**: In Umple, any needed user code can be injected into

the master Umple files; nonetheless, generated code can be easily read as described in Section 3.2.

- **Real-time developers**: There are several UML profiles such as Marte and Autosar for real-time use, but these are hard to master. Umple's C++ code generation (supporting various platforms) and syntax for active objects, ports and composite structure are designed to simplify basic real-time system generation.

Other open source modeling tools are available. ArgoUML [7] was once a contender but has never had full-fledged code generation, and its development has trickled to a very slow pace. Papyrus [8] is an actively-developed open-source modeling suite (According to Ohloh –Black Duck Open Hub [9] its velocity and size is about twice that of Umple), but it is tied tightly to the Eclipse ecosystem, and is more complex than what we desire for our targeted users.

3 The MDE and modeling challenges that Umple addresses

The key challenge Umple addresses is to make modeling simple and adoptable, and hence accessible to most developers. Recent papers have commented on the lack of use of modeling in practice [10], and the obstacles to adoption of modeling [11]. Umple specifically targets these obstacles as described in the following subsections:

3.1 Textual modeling that blends into code and avoids round-tripping

Although many aspects of a system can be better understood using a diagram, textual formats have advantages: They allow rapid input and editing, they allow easier version-difference analysis, and the majority of targeted users are most comfortable with textual forms. We have therefore sought ways to make all modeling constructs textual, and to ensure they are syntactically compatible with our target programming languages. Umple is not the only textual modeling tool, but it is the only tool to allow transparent blending of models with multiple programming languages.

3.2 High quality code generation

Most tools we have studied either do not generate code at all, or else do it in a half-hearted way. It is common that UML associations only generate stub methods [12].

Much Umple research has focused on ensuring generated code code is of top quality and can be used for real systems out of the box. All aspects of Umple-generated code work synergistically with other aspects, and with hand-written code.

Although it is Umple philosophy to never edit generated code, Umple generates readable code. Comments in Umple source pass through to generated code, and traceability links are injected; this enables certification and raises confidence in code correctness. There are thousands of test cases verifying all aspects of the code generation.

3.3 A highly-usable user interface

In a recent paper, we explained how for the education community, Umple's design was guided by the need to achieve usability, incrementality in learning how to model, and various other traits [13]. UmpleOnline instantly starts on the web, and generates code with one click, and diagrams with zero clicks. Umple's command-line tool works just like any other compiler that people have been familiar with for decades, and Umple's Eclipse plugin works just like any other language plugin for Eclipse.

4 Methodology for using Umple

Umple gives the user the freedom to choose their methodology. Virtually any existing approach is possible.

Umple can be used in any of the following modes, or in a hybrid of these:

Model-first: The developer can start by creating the model (either graphically or textually). Developers can then inject any necessary additional program code, such as main programs or methods for algorithms, directly into the Umple text. It is possible in Umple to specify alternative versions of code in different languages. One model can hence be used to create a C++ and a java version of the same system.

Code-first: An existing system written in a pure programming language such as Java or C++ can be 'umplified' [14]. This can be done incrementally in a series of refactorings, gradually adding Umple syntactic constructs to replace existing code. We have so far performed this on systems such as JHotDraw [15] and Weka [16].

5 Research and Development of Umple

Umple has been under development since 2007, and has been the subject of several theses and many published papers that are referenced throughout this paper.

The effectiveness of Umple has been evaluated in several contexts. For example in an experiment [17], Badreddin et al. show that developers can model with Umple's textual form just as readily as they can use the standard UML diagram form for the same model. We plan to conduct more such experiments soon.

One of the key tests of Umple is that it is developed in itself. The Umple compiler code is written in over 120 Umple files, describing over 460 classes. The project is managed using model-driven and test-driven development, as well as continuous integration. The status of the build server [18], and the most recent test run can be found online [19].

Development velocity has been increasing over the years. Most contributions have been by professors and students at ten Canadian and three US universities.

6 Conclusion

Umple is an open-source modeling suite designed to make modeling practical and accessible to a wide variety of software developers and application types.

The accompanying video (http://youtu.be/xD-zTpB_zyQ [20]) gives a walkthrough of the use of UmpleOnline for editing models, generating code and analyzing models. It also gives a quick look at the extensive user manual [2] and the architectural diagram generated by Umple of Umple itself [21]. At the Models conference the demonstration will expand on many of these aspects.

References

1. UmpleOnline, http://try.umple.org
2. Umple user manual, http://manual.umple.org
3. Badreddin, O, Forward, A., and Lethbridge, T.C. (2013), "Improving Code Generation for Associations: Enforcing Multiplicity Constraints and Ensuring Referential Integrity", SERA 2013, Springer SCI 496, pp. 129-149, DOI: 10.1007/978-3-319-00948-3_9
4. Badreddin, O., Lethbridge, T.C., Forward, A., Elasaar, M. Aljamaan, H, Garzon, M. (2014), "Enhanced Code Generation from UML Composite State Machines", MODELSWARD 2014, Portugal
5. Autosar, http://www.autosar.org
6. Aljamaan, H., Lethbridge, T.C., Badreddin, O., Guest, G., and Forward, A. (2014), "Specifying Trace Directives for UML Attributes and State Machines", Modelsward 2014
7. ArgoUML, http://argouml.tigris.org
8. Papyrus, http://www.eclipse.org/papyrus/
9. The Umple Open Source Project on Black Duck Open Hub, http://www.openhub.net/p/Umple
10. M. Petre, "UML in Practice", ICSE 2013, pp 722-731
11. Forward, A., Badreddin, O., and Lethbridge T.C. (2010), "Perceptions of Software Modeling: A Survey of Software Practitioners", 5th C2M:EEMDD Workshop, Paris, June 2010, http://www.esi.es/modelplex/c2m/papers.php.
12. Forward, A. (2010): The Convergence of Modeling and Programming: Facilitating the Representation of Attributes and Associations in the Umple Model-Oriented Programming Language, PhD Thesis, UOttawa, http://www.site.uottawa.ca/~tcl/gradtheses/aforwardphd/
13. Lethbridge, T.C. (2014), "Teaching Modeling Using Umple: Principles for the Development of an Effective Tool", CSEE&T 2014, IEEE Computer Society, Austria, pp 23-28
14. Lethbridge, T.C., Forward, A. and Badreddin, O. (2010), "Umplification: Refactoring to Incrementally Add Abstraction to a Program", Working Conference on Reverse Engineering, Boston, October 2010, pp. 220-224
15. JHotDraw, http://sourceforge.net/projects/jhotdraw/
16. Weka, http, //sourceforge.net/projects/weka/
17. Badreddin, O., Forward, A., and Lethbridge, T. "Model Oriented Programming: An Empirical Study of Comprehension", Cascon, ACM (2012)
18. Umple Continuous Integration Server, http://cc.umple.org
19. Umple Quality Assurance Report, http://qa.umple.org
20. YouTube, Umple Demo – Summer 2014, http://youtu.be/xD-zTpB_zyQ
21. Umple Metamodel, http://metamodel.umple.org