A Tool for Efficiently Reverse Engineering Accurate UML Class Diagrams

Michael John Decker[1], Kyle Swartz[1], Michael L. Collard[2], and Jonathan I. Maletic[1]

[1]Department of Computer Science
Kent State University
Kent, Ohio USA
{kswartz6, mdecker6, jmaletic}@kent.edu

[2]Department of Computer Science
The University of Akron
Akron, Ohio USA
collard@uakron.edu

***Abstract*—A tool that reverse engineers UML class diagrams from C++ source code is presented. The tool takes srcML as input and produces yUML as output. srcML is an XML representation of the abstract syntactic information of source code. The srcML parser (srcML.org) is highly scalable, efficient, and robust. yUML is a textual format for UML class diagrams that can be easily rendered into a graphical diagram via a web service (yUML.me) or a tool such as Graphvis. The approach utilizes efficient SAX (Simple API for XML) parsing to collect the information needed to construct the class diagram. Currently it supports the following UML features: differentiating between class, data type, or interface; identifying design level attributes, multiplicity and type; determining parameter direction; and identification of the relationships aggregation, composition, generalization, and realization. The tool produces yUML for all of Calligra (~1,144KLOC) in under 20 seconds (including translation into srcML). The tool is open source under a GPL license and available for download at srcML.org.**

Keywords—reverse engineering, UML class diagrams, yUML, srcML

I. Introduction

UML (Unified Modeling Language) class diagrams are commonly used in industry for forward engineering. Despite the clear advantages of having UML class diagrams for the maintenance and evolution of a system (e.g., ability to reason about the design of a system at a high-level), these artifacts are rarely maintained or may no longer exist. Several reasons exist for the lack of adoption by industry for maintenance/evolution activities. Among these are, the manual recovery of UML class diagrams is a timely and costly operation. Additionally, most automatic reverse-engineering tools perform poorly, focusing on producing simple class diagrams while failing to properly represent design abstractions [1]. It has also been noted [2] that if tools fail to disclose the internal processes for producing the UML, then this can lead to results that do not meet end-users expectations.

The work of Sutton and Maletic [2] addresses these problem by defining a set of accurate mappings for reverse-engineering of UML class diagrams from C++ source code. The mappings utilize both C++ syntactic and semantic information, as well as, domain knowledge of program conventions, idioms, and reuse libraries to arrive at more accurate UML class diagrams. In a comparison study with several other reverse-engineering tools, their mappings produce comparable diagrams while also better reflecting the abstract design of a system rather than simply recreating implementation-level structures of the program. Another major problem with many reverse-engineering tools is their efficiency and scalability. For example, the most time-efficient approach in the comparison took close to two minutes to recover the UML class diagram from a medium sized project (88 KLOC), while many of the others took 10-20 minutes [2].

In order to provide a useable and available open-source tool that implements these mappings [2], address issues with efficiency, and to provide for a more convenient output format, we created the tool *srcYUML*. The tool uses srcML, an XML representation of source code that combines the textual features of source code with abstract syntax information. It also utilizes extremely efficient SAX (Simple API for XML) parsing to implement the mappings and quickly generate accurate UML class diagram. We choose the UML class diagram text format yUML [3]. The yUML output provides a versatile format that can be rendered online by a web service [3] or via any number of image formats using Graphviz [4]. As the format is fairly simple, it can also be easily parsed and used for static analysis.

Whereas, the most efficient tool studies take approximately two minutes on a medium sized project of 88KLOC, *srcYUML* produces the UML class diagram of a large project (~1,144KLOC) in ~20 sec (including translation to srcML).

II. The srcYUML Tool

srcYUML is a tool written in C++ for reverse engineering UML class diagrams from C++ source code (select files or full project) that has been converted into srcML. Briefly, srcML is an XML representation of source code with the lexical tokens (e.g., identifiers, keywords, operators, etc.) wrapped with information from an AST (Abstract Syntax Tree). The srcML Toolkit allows for extremely fast conversion of source code projects to srcML (92 KLOC/sec) while also preserving the original source code [5]. Once a project (or any amount of subject code) is converted to srcML, *srcYUML* can be used on the srcML to generate the UML class diagram in yUML.

As srcML is an instance of XML, any standard XML technologies (e.g., XSLT, XPATH, DOM, SAX) can be applied. For *srcYUML*, srcSAX [6], a SAX (Simple API for XML) library especially designed for srcML and based off of libxml2 [7], is used to allow for extremely efficient and scalable XML parsing. The efficiency of SAX parsers, both memory and time, is due to the event driven architecture and that SAX parsers do not store information on previously seen tags. To take full advantage of the efficiency of srcSAX, only the necessary information for the diagrams and mappings is gathered (e.g., class names, member variables, etc.). Then,

srcYUML uses the gathered information in conjunction with the mappings in [2] to generate the UML class diagram as yUML, i.e., the classes and their attributes, operations, and relationships. The end result is a UML class diagram that better reflects the abstract design of a system instead of just recreating the base implementation level details.

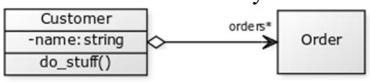
Currently, *srcYUML* supports the following features:

- Separate class types: class, and interface (with datatype to be completed soon)
- Attribute details: multiplicity, type, and visibility
- Operation details
- Parameter details including directionality
- Associations including generalization, aggregation, composition, and realization

The output is in a text format for UML class diagrams called yUML. The website [3] for the format's creator contains details of the syntax and a service to render diagrams. The following is a simple example adapted from yUML.me:

```
[Customer|-name:string;|do_stuff()]
[Customer]<>-orders*>[Order]
```

In brief, UML classes are defined within square brackets (i.e., `[ ]`) with class name, attributes, and operation sections separated by vertical bars (i.e., `|`) respectively and individual attributes and operations separated by semicolons. The attributes and operations are specified as Unicode text. In the above example, the first line is a single class with name `Customer`, member `name` of type `string`, and method `do_stuff`. An association is specified between UML classes with dash `-` for a solid line and `-.-` for a dashed line. Names and multiplicity are added to the left and/or right as Unicode text. Directionality is added by specifying the appropriate arrow (`<` or `>`). The type of association can be further defined by using `++` for composition, `<>` for aggregation, and `^` for generalization. The second line of the example shows an aggregation between `Customer` and `Order` along with name and multiplicity. The following is the rendered version of the yUML example:



yUML supports additional features we do not describe here. The reader is referred to yUML.me for more details on the format. In essence, yUML is a simple output format that is easily parsed, can be rendered simply using yUML.me's web service, or can be rendered locally via Graphviz. We are currently working on and will soon provide local rendering utilizing ANTLR [8] for parsing of the yUML and generated images using Graphviz's *dot* language and tool. Since, yUML is easily parsed and contains information about the design-level of a system, this information can also be used as a convenient format for static analysis. The yUML.me web service has limitations and some characters such as brackets cannot be used inside UML classes (e.g., for multiplicity). Typically, Unicode characters can be used instead (e.g., `[` and `]`). We plan on removing this limitation for our local rendering; however, currently *srcYUML* outputs Unicode characters so that the web service can be used. Lastly, the web service can have difficulty rendering large diagrams. Local rendering will alleviate any such problems.

srcYUML is extremely efficient. The entire Calligra project [9] is rendered from srcML into yUML in ~6.5 seconds on an iMac running El Capitan with 2.7 GHz Intel Core i5 processor and 8 GB DDR3 RAM. With the translation to srcML taking less than 12.5 seconds, the UML class diagram for all of Calligra (~1,144KLOC and over 5,500 classes) can be created directly from source in under 19 seconds.

III. Using srcYUML

srcYUML is run as a command-line tool as follows (currently, only UNIX-like operating systems are supported):

```
srcyuml srcml_input.xml output.txt
```

The first argument *srcml_input*.xml is a srcML archive (collection of files or project in srcML) and the second argument is the file in which to output the reverse engineered yUML. The tool is available at srcML.org in the downloads page or directly from the git repository (consult the README.me for instructions) [10].

III. Related Work

Commonly, reverse engineering of UML diagrams is supported via the addition of reverse-engineering parsers and analyses to existing UML modeling tools. Examples are Rational Rose, Umbrello, and Visual Paradigm. For a more complete list of UML Modeling and whether they support reverse-engineering see [11]. The C++ mappings were previously implemented in an in-house research prototype called *pilfer* [2]. That approach used an inefficient, non-scalable DOM (Document Object Model) approach and was never made publicly available. In addition, it did not output yUML. *srcYUML* replaces *pilfer* as a publicly-available, open-source, and highly efficient tool that outputs in yUML. For additional related work on reverse engineering mappings and the pilfer tool, please see [2].

IV. Conclusion and Future Work

We present *srcYUML* a highly efficient and accurate tool for reverse engineering class diagrams. The tool uses mappings as defined in [2] to arrive at accurate UML class diagrams which better reflect the abstract design of a system.

In order to progress the project into maturity, we will work on improving the tool both in the quality of the reverse-engineered UML class diagrams (abstract design and otherwise) and adding new features. A few of the features will be to add local rendering via Graphviz and provide/experiment with different layout algorithms. Furthermore, since srcML supports other languages (e.g., Java and C#), we can apply techniques similar to [2] for those languages, and then add support for those languages to *srcYUML*. By supporting additional languages, *srcYUML* will also allow modeling of multi-language projects. Although srcYUML abstracts out some implementation details, we will also investigate support for developers who are less interested in using UML as a blueprint and more interested in using it to grasp the most important details of the design. Future work, in this regard, will consist of automatically identifying what details of the UML are best displayed to users and providing an option for user to define what areas/classes/etc. is of the most interest.

Acknowledgment

This work was supported in part by a grant from the U.S. National Science Foundation CNS 13-05292/05217.

References

[1] Kollman, R., Selonen, P., Stroulia, E., and Zündorf, A., "A Study in the Current State of the Art in Tool-supported UML-based Static Reverse Engineering", in Proceedings of Ninth Working Conference on Reverse Engineering (WCRE'02), Richmond, Virginia, Oct 29-Nov 1 2002, pp. 22-34.

[2] Sutton, A. and Maletic, J. I., "Recovering UML Class Models from C++: A Detailed Explanation", *Information and Software Technology*, vol. 49, no. 3, Jan 2007 2007, pp. 212-229.

[3] yUML, Date Accessed: June 27, 2016, http://yuml.me,

[4] Graphviz, Date Accessed: June, 27, 2016, http://www.graphviz.org,

[5] Collard, M. L., Decker, M. J., and Maletic, J. I., "srcML: An Infrastructure for the Exploration, Analysis, and Manipulation of Source Code: A Tool Demonstration", in Proceedings 2013, pp. 516-519.

[6] srcSAX, Date Accessed: June 27, 2016, 1 https://github.com/srcML/srcSAX,

[7] libxml2, Date Accessed: June 27, 2016, http://xmlsoft.org,

[8] ANTLR, "The ANTLR Translator generator", Web page, Date Accessed: June 27, 2016, http://www.antlr.org/, 2014.

[9] Calligra, Date Accessed: June 27, 2016, https://github.com/KDE/calligra.git,

[10] srcYUML, Date Accessed: June 27, 2016, https://github.com/srcML/srcYUML,

[11] Wikipedia, "List of Unified Modeling Language Tools", Date Accessed: June 27, 2016, https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools, 2016.