CPP2XMI: Reverse Engineering of UML Class, Sequence, and Activity Diagrams from C++ Source Code

E. Korshunova[1], M. Petković[1], M.G.J. van den Brand[1], M.R. Mousavi[2]
[1]Laboratory for Quality Software, [2]Technische Universiteit Eindhoven,
Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{e.korchounova, M.Petkovic, M.G.J.v.d.Brand, M.R.Mousavi}@tue.nl

1. Introduction

The Unified Modeling Language (UML) is the de-facto standard in object-oriented software development. Although mainly used during the design phase, UML models can still be valuable after the delivery, when a system enters the maintenance phase. However, for many software systems only the source code and possibly very limited and inconsistent documentation is available. In order to fully understand an existing object-oriented system (e.g., a legacy system) that is potentially incomplete, information regarding its structure, behavior, and internal states has to be extracted from the source code and must be represented in the form of an abstract (e.g., UML) model. The process of analyzing the source code to represent it at a higher level of abstraction, by extracting architectural artifacts from the code, is called *reverse engineering.*

In the context of this work, reverse engineering is used as a part of the *verification and validation* chain of software systems, where the static structure and the dynamic behavior of the system are derived from the source code and represented in XML Metadata Interchange (XMI) format. The obtained model is further analyzed for such characteristics as soundness and complexity of the system. XMI [4] is a standard that enables to express objects using Extensible Markup Language (XML). XMI can be used to represent objects from UML model in XML.

In this paper, we describe the reverse engineering tool, CPP2XMI, which allows extracting UML class, sequence, and activity diagrams in XMI format from C++ source code, and its position in the toolset for software system analysis.

2. CPP2XMI – a reverse engineering tool

Most Computer Aided Software Engineering (CASE) tools can reverse engineer class diagrams while there is little tool support for extracting sequence or activity diagrams from C++ source code. Therefore, we have developed a reverse engineering tool called CPP2XMI for transforming C++ source code into UML class, sequence, and activity diagrams. We decided to use as much of the existing technology as possible, and thus exploited existing tools and standards, such as the Columbus/CAN fact extractor [1], XMI [4], and DOT [2].

Fig.1 shows an elaborated architecture of CPP2XMI. This architecture divides the tasks of a system into several sequential processing steps. Each processing step is encapsulated in a separate module, represented as an oval in Fig.1.

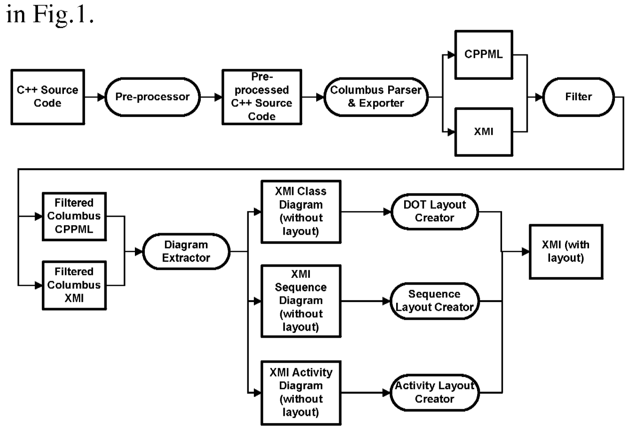

Figure 1. Elaborated architecture of CPP2XMI

The main modules of the system are the following:

1) Columbus Parser and Exporter

Columbus/CAN [1] is a fact extractor that offers functionality for parsing source code and exporting the generated Abstract Syntax Tree (AST) into different formats. Two of them are of interest for our project: *UML XMI* (v.1.1) and *C++ Markup Language (CPPML).* The UML XMI output contains all information about class diagrams, including classes and relations between them. The CPPML output is an XML formatted file that contains all the information from the AST including detailed information from the method bodies. It can be used to generate UML sequence and activity diagrams.

2) Filter

The Columbus output is huge because it includes information from standard C++ libraries, which is not relevant for UML diagram creation. Therefore, we have developed the *Filter* module that, by removing the redundant information, reduces the Columbus output size significantly. This, in turn, saves extra effort from the subsequent modules and enhances the readability of the outcome as well as the performance of our tool.

3) Diagram Extractor

The main purpose of the *Diagram Extractor* module is to perform a transformation from filtered output of the parser into the format suitable for UML v.1.3 diagram representation, i.e., the XMI v.1.1 format, in our case.

The XMI creation process used by the *Diagram Extractor* module is shown in Fig.2.

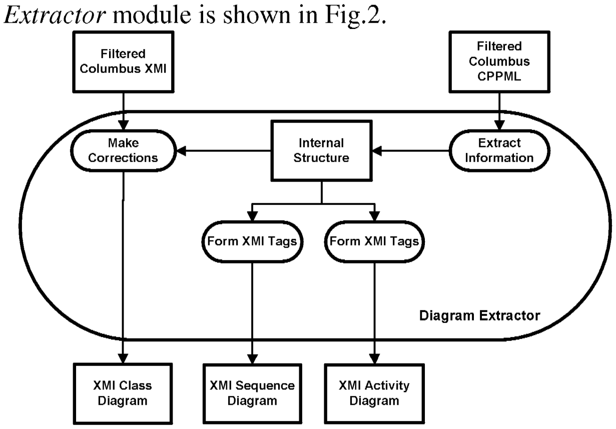

Figure 2. XMI creation using Columbus output

First, we extract complete information about the code, including objects allocated in the program and function calls from the Columbus CPPML output. We need this information to create XMI elements for the sequence diagram.

Furthermore, we extract information from conditional and iterative statements, which is important for activity diagram generation.

All retrieved information is stored in the internal structure.

After the internal structure is created, we correct the Columbus XMI output. We cannot use it directly, because it has certain defects that we will not discuss in this paper.

Finally, we create XMI tags for sequence and activity diagrams.

4) Layout Creator

The main purpose of the *Layout Creator* module is to automatically generate layout for UML diagrams, in order to visualize them with the in-house visualization tool MetricView [3] or other CASE tools.

We use the DOT [2] tool (part of the Graphviz framework) for the visualization of UML class diagrams.

To the best of our knowledge, no existing tool allows generating coordinates of *objects* and *messages* in the sequence diagram. Some tools sketch sequence diagrams, but store them only in PNG or EPS formats. Therefore, to generate sequence diagram layout, we developed our own algorithm that determines coordinates for *objects* and *messages* communicated between them.

3. CPP2XMI as the part of the software analysis toolset

The availability of CPP2XMI closed the gap in our toolset called SQuADT [8] used for the analysis of C++ source code. This toolset consists of the metric derivation tool, called SAAT [5], and metric visualization tool, called MetricView [3]. The SAAT tool takes UML model in the XMI format as its input and generates metrics for it. Some of the metrics can be visualized on top of the UML class and sequence diagrams by the MetricView CASE tool. MetricView can open UML class and sequence diagrams in the XMI format and show metrics generated by SAAT in 3D and 2D views. The metrics can help to find out and to argue about possible flaws in the architecture that is extracted from the source code.

Besides the analysis of UML class and sequence diagrams, we also performed transformation of UML activity diagrams into Petri Net models. Petri Nets can be further analyzed by Woflan [6] or mCRL2 [7] tools for checking such properties as the soundness of the system.

In order to prove an industrial value of the described techniques, we have performed the analysis of two large-scale case studies: one of about 30000 lines and another one of about 60000 lines of C++ code.

References

[1] Ferenc, R., Beszédes, Á., Tarkiainen, M., and Gyimóthy, T. "Columbus - Reverse Engineering Tool and Schema for C++". In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM 2002)*, pp. 172-181, IEEE CS, 2002.

[2] Gansner, E.R., Koutsofios, E., North, S.C., and Vo, K.-P. "A Technique for Drawing Directed Graphs". *IEEE Transactions on Software Engineering*, 19:214-230, 1993.

[3] Termeer, M., Lange, C.F.J., Telea, A., and Chaudron, M.R.V. "Visual Exploration of Combined Architectural and Metric Information". In *Proceedings of VISSOFT 2005,* IEEE CS, 2005.

[4] Grose, T.J., Doney, G.C., and Brodsky, S.A. *Mastering XMI: Java Programming with XMI, XML, and UML*, John Wiley and Sons, 2002.

[5] Lange, C. F. J., *Empirical Investigations in Software Architecture Completeness*, Master's Thesis, Eindhoven University of Technology Press, 2002.

[6] Woflan *home page*, http://is.tm.tue.nl/research/woflan/index.htm.

[7] mCRL2 *home page,* http://www.mcrl2.org/.

[8] SQuADT *home page*, http://www.laquso.com/research/repository.php.