This paper has been accepted at ICSM 2004 as poster.

Abstract and Precise Recovery of UML Class Diagram Constituents

Yann-Gaël Guéhéneuc
Département d'informatique et de recherche opérationnelle
Université de Montréal – CP 6128 succ. Centre Ville
Montréal, Québec, H3C 3J7 – Canada

E-mail: guehene@iro.umontreal.ca

Abstract

Reverse-engineered UML class diagrams are neither abstract nor precise representations of source code because of the loose definitions of UML constituents. Thus, they are of little interest for software maintainers. We perform an exhaustive study of UML class diagrams constituents with respect to their recovery from C++, Java, and Smalltalk source code. We implement a tool suite, PTIDEJ*, to reverse engineer Java source code abstractly and precisely.*

1 Problem

UML class diagrams produced during design are often forgotten during implementation, under time pressure usually. Thus, they present major discrepancies with implementation frequently and are of little help to maintainers who must support released programs.

Maintainers need means to recover UML class diagrams from programs implementation. These means must be *automated* considering the large size of programs and they must produce *abstract* yet *precise* class diagrams to help maintainers in their tasks.

2 Proposed Solution

We survey the literature on programs reverse-engineering to find suitable criteria to evaluate reverse-engineered UML class diagrams with respect to their usefulness for maintainers [1, 2, 4]. We propose two criteria: Abstractness and preciseness.

We survey existing reverse-engineering tools and other tools with reverse-engineering capabilities, such as CHAVA [5], ARGOUML, IDEA, Rational ROSE, Together TOGETHERJ, WOMBLE [3]. We show that these tools do not produce abstract nor precise class diagrams with respect to source code.

We perform an exhaustive study of thirty-four UML class diagram constituents from the UML v1.5 specifications, Chapter 3, Part 5 "Static Structure Diagrams", sections 3.21 through 3.53 [6]. We assess the abstract and precise automated recovery of these constituents from source code constructs in C++, Java, and Smalltalk. We show that many of these constituents can indeed be abstractly and precisely recovered from source code automatically. This study is also a step towards unambiguous definitions of UML constituents through correspondence with source code.

3 Status and Future Work

We implement the results of our study in our tool suite, PTIDEJ, for the abstract and precise reverse-engineering of Java programs. We perform some experiments using our tool suite on different programs.

Future work includes enhancing the implementation and performing concrete maintenance tasks to assess the usefulness of abstract and precise reverse-engineered UML class diagrams.

References

[1] B. Bellay and H. Gall. A comparison of four reverse engineering tools. In *proceedings of WCRE*, pages 2–11. IEEE CS Press, Oct. 1997.
[2] G. C. Gannod and B. H. C. Cheng. A framework for classifying and comparing software reverse engineering and design recovery techniques. In *proceedings of WCRE*, pages 77–88. IEEE CS Press, Oct. 1999.
[3] D. Jackson and M. C. Rinard. Software analysis: A roadmap. In *proceedings of ICSE*, pages 133–145. ACM Press, Jun. 2000.
[4] R. Kollmann, P. Selonen, E. Stroulia, T. Systä, and A. Zündorf. A study on the current state of the art in tool-supported UML-based static reverse engineering. In *proceedings of WCRE*, pages 22–33. IEEE CS Press, Oct. 2002.
[5] J. Korn, Y.-F. Chen, and E. Koutsofios. Chava: Reverse engineering and tracking of Java applets. In *proceedings WCRE*, pages 314–325. IEEE CS Press, Nov. 1999.
[6] Object Management Group, Inc. *UML v1.5 Specification*, Mar. 2003.