Experiences in Teaching Program Transformation for Software Reengineering

Mohammad El-Ramly
mer14@le.ac.uk
Department of Computer Science,
University of Leicester, UK

Extended Abstract

Little attention is given to teaching the theory and practice of software evolution and change in software engineering curricula. Program transformation is no exception. This paper presents the author's experience and lessons learned from teaching program transformation, particularly source-to-source transformation, as a technique for software reengineering in a postgraduate module on software systems reengineering.

The module was taught as part of the M.Sc. of Software Engineering for the e-Economy at University of Leicester in 2003/2004 and 2004/2005. In the 2003/2004 offering, Turing eXtender Language (TXL) was used for teaching the program transformation unit. TXL is a programming language specifically designed for source-to-source transformation. It takes as input an arbitrary context-free grammar EBNF notation and a set transformation rules. TXL parses the input programs in the language described by the grammar, no matter if ambiguous or recursive, and then successively applies the transformation rules to the parsed input until they fail. Then, it unparses the transformed parse tree to output the transformed source. Gradually, students built their TXL skills until they could write a grammar for a limited language that only has variable declarations, assignments, 'if', 'case', 'while', 'goto' statements and labels. Then, they could write a TXL clone detector or a TXL program-restructuring ('goto' elimination) tool for programs in this language.

In 2004/2005, code transformation was taught in a condensed 12-hours industrial tutorial by an industry expert from ATX Software using ATX's Legacy-CARE (or L-CARE) environment for legacy software reverse engineering and reengineering. The tutorial consisted of lectures, two demos, lab sessions and an assignment. The two big industrial case studies shown were on code certification (detecting violations of a company's coding standards manual) and transforming a huge Cobol system from persistent file storage to database storage. Students learned how to use L-CARE for simple tasks of querying the code using an extension of XPath for patterns of interest and writing simple transformation rules to apply to the detected patterns. L-CARE applies the transformations to the source code directly not to the input's parse or abstract syntax trees.

From these experiences, it was learned that selecting the suitable material (that balances theory and practice) and the right tool(s) for the level of students and depth of coverage required is a non-trivial task. Most of the available material and tools are either research- or industrial-oriented. Little is available for educational purposes.

It was also found that teaching using toy exercises and assignments does not adequately convey the practical aspects of the subject. While, teaching with real, even small size, exercises and assignments, is almost non-feasible. Finding the right balance is very important but not easy. Finally, students' understanding and appreciation of the topic was higher when they were presented with real industrial case studies. Also, Students with industrial experience understood and appreciated the potential and applicability of this technology more than the rest.

Dagstuhl Seminar Proceedings 05161
Transformation Techniques in Software Engineering
http://drops.dagstuhl.de/opus/volltexte/2006/423