A Software Engineering Curriculum Incorporating Formal Methods: A Progress Report

Ann E. Kelley Sobel
Systems Analysis Department
Miami University
Oxford, OH 45056

Abstract - This project, which is funded by the NSF Educational Innovation Program, integrates a formal method into the undergraduate software engineering curriculum of the Systems Analysis Department of Miami University. A formal method consists of a mathematical model and methodology to aid in the creation of the specification, design, and verification of a software system. The justification for the introduction of a formal method not only stems from government and industry regulations/standards, but also from the national perception that the discipline of software engineering education lacks sufficient emphasis on mathematics and engineering science.

Interest exists among faculty to introduce formal methods into individual undergraduate courses, however no attempts have been made to base several standard core curriculum courses on learning and applying formal methods. Teaching undergraduates material that is typically reserved for graduate students poses special problems and challenges for the instructor. The approach taken to overcome these difficulties include frequent attention to student motivation, student mentor support, the use of a basic, operationally-based formal method, and the application of this formal method throughout the student’s academic experience.

The experimental course track integrates formal method techniques into the core curriculum courses of data structures, software engineering principles, and systems analysis and design; culminating in a senior capstone project. The experimental sequence also includes two new courses for teaching the skills necessary to apply formal methods. Formal methods and their application in programming and software system creation were introduced at the sophomore level to ensure substantial use and training. This application culminates in a one year senior project where student project teams work with an industrial partner to generate a software system where formal analysis is applied throughout the phases of software development.

For each semester, instructors are responsible for teaching both the modified and regular course sections. As the experimental and control group of students progress through the core curriculum in parallel, we are able to directly assess the differences in student analytical and problem solving skills as well as student’s perceived benefits/disadvantages.

Sixteen freshmen began this modified sequence of courses with a discrete mathematics course in the Spring semester of 1996. The basic skills needed to apply formal methods were presented in a new course, Logical Foundations of Programs, in the Fall semester of 1996 which covered mathematical induction, well-foundedness, specification, derivation, and the theory of the correctness of programs. The text of Cohen, Programming in the 1990s, was chosen because the presentation of this material was not coupled with an implemented programming language. Gaining experience in performing logical deduction was motivated by assigning Smullyen logic puzzles. Classroom instruction was supplemented with several in-class problem solving sessions which were supported by three undergraduate mentors. By the end of the semester, students were able to derive algorithms and verify that their algorithm satisfied the first-order logic specification.

The next course in the experimental sequence, Data Abstraction and Structures, offered Spring 1997, focused on implementing abstract data types while developing programs written using the C++ programming language. This course introduces both procedural and data abstraction using standard data structures and the algorithms for manipulating them. Correspondence between abstract object operations and the chosen implementation type operations were maintained using coupling invariants. Multiple abstract data type implementations were verified for a variety of the standard data structures: lists, stacks, queues, and binary trees.

Comparisons of student performance on exams and clinical interviews will be reported.