An Evolution of a Software Engineering Curriculum

Colin Atkinson, David Eichmann and Charles McKay

University of Houston - Clear Lake
http://ricis.cl.uh.edu/SE

Abstract. Many software engineering master's programs, particularly in the U.S., are based on the SEI curriculum archetype published in 1987. This paper describes how one such program at the University of Houston - Clear Lake is being modified to accommodate developments in software engineering practice and to better meet the needs of a particular local constituency. Highlights of the revised program include a new 5-course core curriculum, four concentration areas and a two-semester capstone experience enabling students to apply their knowledge to real-world projects. The new curriculum is designed to provide an optimum balance between the needs of students, the requirements of industry and the integration of new technology.

1 Introduction

Software engineering became recognized as a legitimate academic discipline only about 10 years ago. Since that time the majority of software engineering programs, at least in the U.S., have been based on the curriculum archetype developed by the Software Engineering Institute (SEI) in 1987. The master's degree in Software Engineering at the University of Houston - Clear Lake (UHCL) is now four years old and is still largely based on the SEI curriculum. While the SEI curriculum archetype has been highly successful in spawning new software engineering programs, we believe it is time for many of these programs to evolve from this common starting point and respond to recent advances in the body-of-knowledge, practice and technology of software engineering. Chief among these advances are:

1. the increasing importance of process modeling, control and improvement;
2. the explosion in the manipulation, storage and communication of information, particularly in connection with the so-called information superhighway;
3. the rising problem of legacy software systems and the associated reengineering requirements;
4. the increasing importance of formal methods and notations;
5. the use of software for ever more demanding and critical applications; and
6. the spectacular success of the object-oriented paradigm and the move towards methods which cover the full development cycle.
In the past, UHCL faculty have responded to these developments primarily by offering specialized ‘topics’ courses that tackle individual subjects in isolation. However, we believe software engineering has now reached a level of maturity where it is appropriate to integrate these advances throughout the curriculum and to address the various specialized topics and technologies in terms of ‘concentration areas’ or ‘tracks.’ Among other things, this implies a curriculum which embodies a new hierarchy of concepts, principles and courses, and which gives students more guidance in terms of class sequencing and grouping. To this end, the software engineering faculty at UHCL have been working over the past year on a new curriculum that will integrate the above technology advances in a way that provides optimum flexibility to students and more closely meets the needs of the community which UHCL serves. The work has primarily focused on the development of:

1. four new concentration areas or tracks;
2. a new five-course core curriculum; and
3. a new two-semester capstone experience.

This paper provides an overview of the new program under development at UHCL. As a starting point we first describe the SEI curriculum archetype and discuss some of its strengths and weaknesses. We then go on to describe the current UHCL packaging of this curriculum in section 3 before introducing the new curriculum in section 4.

2 The SEI Archetype

In 1987 the SEI published a set of curriculum ‘units’ which were intended to lay the foundation for new software engineering programs across the country [5]. Units could be combined and sequenced to support a number of courses. The UHCL packaging of these units into courses is believed to be fairly representative of other programs [6]:

1. Specification of Software Systems  
2. Principles and Applications of Software Design  
3. Software Generation and Maintenance  
4. Software Engineering Process  
5. Software Verification and Validation  
6. Software Project Management  
7. Formal Methods and Models

To help institutions ‘bootstrap’ the program, many of these curriculum units were supported by video tapes and other ‘courseware.’ Professors teaching a course for the first time were able to introduce the courses with the help of these tapes and materials.

Although this approach proved very successful in establishing new software engineering programs, we believe it also has some shortcomings: