Developing Undergraduate Software Engineering Programs

Mark J. Sebern
EECS Department
Milwaukee School of Engineering
1025 North Broadway
Milwaukee, WI 53202-3109
sebern@msoe.edu

Michael J. Lutz
Software Engineering Department
Rochester Institute of Technology
134 Lomb Memorial Drive
Rochester, NY 14623-5608
mikelutz@mail.rit.edu

Abstract
This workshop provides a forum for discussing issues related to the development of undergraduate software engineering programs. Participants discuss means by which software engineering concepts can be incorporated in either new or existing baccalaureate programs.

1. Workshop objectives
The objectives of the workshop are: (1) to report on the experiences of some institutions now offering or preparing to offer undergraduate software engineering programs, (2) to identify issues that affect the development and operation of these programs, and (3) to foster dialog and to promote cooperation in the development of these programs.

2. Organization and format
Each participant will present a brief opening statement. Attendees will then be polled regarding the areas and issues in which they have the greatest interest. The facilitators will guide a structured discussion of the selected topics.

3. Final deliverable
The workshop activity will be summarized in a report, including a directory of known existing and planned undergraduate software engineering programs. The report will be distributed and shared with individuals and organizations interested in the development of undergraduate software engineering programs.

4. Position statements of the workshop facilitators
4.1. Undergraduate software engineering in an engineering context (Sebern)
Each undergraduate software engineering program is influenced by the context in which it develops. Some may grow out of an existing computer science program, while others are sponsored jointly by existing computer science and engineering units either within or across college boundaries. A software engineering program may also evolve in the context of an existing computer engineering program with significant software content, the situation at the Milwaukee School of Engineering (MSOE).

A private university founded in 1903, MSOE offers undergraduate programs in architectural, biomedical, computer, electrical, industrial, and mechanical engineering, but not in computer science. The computer engineering program gives equal weight to hardware and software content. In recent years, a number of students expressed a desire for a program with more software content, while retaining a focus on engineering concepts,
methods, and process. At the same time, many industrial partners found that their
businesses were becoming ever more dependent on software. This combination of student
“push” and industry “pull” motivated the development of the software engineering program
which, after more than two years of preparation, began operation in fall 1999 with both
freshman and sophomore (internal transfers from computer engineering) students.

MSOE’s software engineering program has a heavy laboratory and project emphasis in
major courses, an application domain elective sequence, and a software development
laboratory with large, ongoing projects, culminating in a team-based capstone senior design
sequence. Individual and team processes are integrated into the curriculum.

Current issues include: (1) tracking the evolving ABET program criteria for software
ingineering, (2) recruiting new faculty, (3) ongoing professional development for current
faculty, and (4) implementation and maintenance of hardware and software infrastructure.

4.2. Key elements in the development of RIT’s B.S.S.E. program (Lutz)

RIT was the first university in the United States to offer a baccalaureate program in
software engineering. Currently in its fourth year of operation, the program prepares
graduates for careers in commercial and industrial software development. The focus of this
report is on the factors that drove development and approval. In retrospect, a few elements
were crucial to gaining final approval; our observations may serve to guide others
developing such programs.

4.2.1. Highly-placed champion: We had a champion who was a graduate of RIT, an
executive vice-president at a large engineering firm, and a member of the Board of
Trustees. His efforts served to make the software engineering effort visible to the Institute’s
administration, and the top-down pressure thus exerted cleared many of the obstacles from
our path.

4.2.2. A committed team, a shared vision: The program was the product of five faculty
members in computer science and computer engineering. Though we had different
backgrounds and experience, we soon agreed on a shared vision of the program. Without
the force of this vision, we might well have ended up with an unfocused program consisting
of a collection of “pet” courses.

4.2.3. Interdisciplinary cooperation: Software engineering is not an island – at the very
least, it builds on the foundations set by computer science and computer engineering. These
two departments supported our efforts, helped set the proper foundations, and provided
cover from external criticism.

4.2.4. Industrial validation: RIT’s primary mission is professional career education; thus a
critical issue was the program’s acceptance by industry. The industrial advisory committee
formed to evaluate the proposal served both to improve the program and, in their
overwhelming enthusiasm, to sway remaining doubters.

4.2.5. Conclusion: RIT’s software engineering program is already successful, even before
the first students graduate. Co-operative education placement rates hover around 100%,
with many employers eager to recruit even more students. This is testimony to the need for
the program; fortunately for us, RIT was willing to gamble on the program’s success. The
gamble is already paying handsome dividends in prestige, recruitment, and enrollment.