CURRICULUM DEVELOPMENT

SE2004: Recommendations for Undergraduate Software Engineering Curricula
by Timothy C. Lethbridge, Richard J. LeBlanc Jr., Ann E. Kelley Sobel, Thomas B. Hilburn, and Jorge L. Díaz-Herrera, pp. 19–26. SE2004 evolved from an analysis of desired student outcomes for a software engineering graduate as compared to those for computer science and computer engineering graduates. This survey of the recommendations summarizes the Software Engineering Education Knowledge, which outlines what a software engineering undergraduate should learn while at a university. The authors also summarize a set of 19 guidelines prepared to help educators implement a software engineering program as well as the courses and a sample of the curriculum patterns that SE2004 proposes.

Creating an Accreditable Software Engineering Bachelor’s Program
by Stephen T. Frezza, Mei-Huei Tang, and Barry J. Brinkman, pp. 27–35. In the light of a developing discipline and evolving accreditation standards, the design of a new bachelor of science in software engineering program can be challenging. This article outlines the requirements sources, effective design processes, and a model curriculum for a BSSE program in a small university.

Learning Software Engineering at a Distance
by Brendan Quinn, Leonor Barroca, Bashar Nuseibeh, Juan Fernández-Ramil, Lucia Rapanotti, Pete Thomas, and Michel Wermelinger, pp. 36–43. The Open University’s Computing Department offers specialized master’s programs in software development and in the management of software projects. The OU characterizes these professionally accredited programs as part-time, open, and large-scale distance learning primarily aimed at IT practitioners. These characteristics guide the development of the software engineering curriculum. Although focused on the OU’s particular experiences, the article also outlines broader lessons learned that might benefit other institutions engaged in designing and delivering software engineering curricula to a geographically distributed student body.

Experiences with Open Source Software Engineering Tools
by Kal Toth, pp. 44–52. Open source software offers a unique opportunity for improving learning outcomes for software engineering and computer science education. This article describes experiences exploiting OSS to enhance the practical aspects of such programs. In addition to lessons learned, the article offers guidance for those wanting to enhance the practical aspects of their practicum and capstone courses.

A Model Curriculum for Aspect-Oriented Software Development
by Johan Brichau, Ruzanna Chitchyan, Siobhán Clarke, Ellie D’Hondt, Alessandro Garcia, Michael Haupt, Wouter Joosen, Shmuel Katz, Jacques Noyé, Awaï Rashid, and Mario Südholt, pp. 53–61. A model postgraduate curriculum for aspect-oriented software development has been derived from the practical experience of running postgraduate courses at nine institutions across Europe (part of the European Network of Excellence on AOSD). Several institutions have made concrete instantiations of the curriculum, from a summer school to a university master’s degree course. The curriculum also relates to the Software Engineering Body of Knowledge.

FEATURES

In Search of What We Experimentally Know about Unit Testing
by Natalia Juristo, Ana M. Moreno, Sira Vegas, and Martín Solari, pp. 72–80. Software engineering is a relatively young field and experimental software engineering is even younger, so undisputed facts are few and far between. Nevertheless, experimental results can help practitioners make better decisions, and the authors have aggregated results derived from individual unit-testing experiments published in high-quality journals and proceedings. Most of the experiments focus on two important characteristics of testing techniques: effectiveness (number of faults found) and efficiency (effort required to apply the technique).

Are CMM Program Investments Beneficial? Analyzing Past Studies
by Daniel Galin and Motti Avrahami, pp. 81–87. Many software development professionals and managers aren’t fully convinced that programs to raise their organizations’ CMM level will provide a good return on investment. Unfortunately, because of the scattered nature of published research on this topic, the results haven’t been conclusive. An investigation of CMM level transitions for seven common performance metrics (including ROI) analyzed a database of more than 400 projects from 19 information sources. This analysis found that CMM program investments improved performance consistently for all seven metrics.

A Model for Technology Transfer in Practice
by Tony Gorschek, Per Garre, Stig Larsson, and Claes Wohlin, pp. 88–95. Successful technology transfer requires close cooperation and collaboration between researchers and practitioners. Researchers need to observe the challenges facing industry firsthand and tailor their work accordingly. Practitioners can help shape technology development on the basis of tangible issues identified on site. This article presents a seven-step technology transfer model that reflects collaborations between university researchers and practitioners at two Swedish companies.