Guest Editor’s Introduction

SOFTWARE engineering education is of widespread and growing interest throughout industrial organizations, governmental agencies, and academic institutions. Industry is interested in software engineering education because economic advantages accrue to organizations that can consistently produce well-engineered software artifacts on time and within budget. Governmental agencies are concerned with the role of software engineering education in maintaining a technological infrastructure that will provide for national security and ensure a competitive position in the international marketplace. Academe is motivated by a desire to serve the needs of modern society and by the intellectual challenges inherent in this emerging discipline.

Because development and modification of computer software are labor intensive activities, better education and training for software engineers are essential elements of better quality and increased productivity. During the past 10 or 15 years, there has been a steady increase in educational activities related to software engineering, within both the industrial and academic communities. Pioneering efforts in software engineering education include the early papers by Freeman, Wasserman, and Fairley [1]–[5]; in-plant training programs within companies such as Lockheed [6]; and the Masters programs at Seattle University, Texas Christian University, and Wang Institute of Graduate Studies. These three Masters programs were inspired by the papers cited above and by a curriculum report produced for the IEEE Computer Society in 1978 by a committee chaired by Fairley [7].

During the past three years, the Software Engineering Institute at Carnegie-Mellon University has emerged as a focal point for activities in software engineering education. The Software Engineering Institute (SEI) is a federally funded research development center that was established by the U.S. Department of Defense to improve the state of software technology. Software engineering education is one of the major programs within the SEI; the charter of the Education Division at SEI is to develop software engineering curricula and supporting materials for use by educational institutions.

In each of the past two years, SEI has hosted a meeting of software engineering educators. Each of those meetings has resulted in a conference proceedings published by Springer-Verlag [8], [9]. The first volume contains 24 papers, summaries of discussions, and reports from working groups. The second volume contains a similar number of papers plus summaries of discussions and panel sessions. The papers presented here are based on papers that were first published in those proceedings. Springer-Verlag has granted IEEE permission to reprint material that might be redundant between the earlier versions and the present versions of the papers.

The ten papers that appear in this issue summarize the current state of software engineering education, present novel approaches, and indicate areas for future efforts. Peter Freeman’s paper revisits a landmark paper published by Freeman, Wasserman, and Fairley in 1976 [1]. He discusses progress made in the interim period and presents some revisions to the recommendations contained in the earlier paper. Mark Ardis discusses the evolution of the Masters program in software engineering at Wang Institute. That program represents the most concerted effort to date in graduate level software engineering education.

The next three papers present three different approaches to teaching software design and implementation. David Weiss relates his experiences in teaching the NRL design methodology to graduate students at Wang Institute. Jon Bentley and John Dallen relate their approach to teaching software design and implementation to undergraduates at West Point, and James Burns and Edward Robertson present a novel approach used at the University of Indiana in which students in a graduate level software engineering course serve as project managers for students conducting software projects in the companion undergraduate course.

The papers by David Wortman and Bill McKeeman describe two approaches to teaching software engineering project courses. It is widely agreed that project courses in which students work in teams and apply good software engineering techniques are an essential component of a software engineering education program. However, there are many different ways to organize and conduct project courses. Based on his experiences during the past several years at the University of Toronto, Wortman presents an update of the famous “Software Hut” paper published in this journal by Horning and Wortman in 1977 [10]. McKeeman describes an approach to teaching software project courses that he has developed and pursued at Wang Institute.

The paper by Leventhal and Mynatt summarizes the results of their survey of undergraduate software engineering courses. The final two papers, one by Bill Riddle and Lloyd Williams, and the other by Jonah Lavi and colleagues present industrially oriented views of software engineering education and training. Riddle and Williams address the issue of continuing education to enable managers and technical personnel to decide whether or not to adopt a particular technological innovation. Lavi and colleagues describe three different programs of study, each of approximately seven months duration, to train employees of Israel Aircraft Industries in software engineering. The three programs are tailored to three application areas of software engineering: embedded real time systems, CAD/CAM applications, and data processing.

The papers in this collection thus present a representative view of software engineering education at the graduate level, the undergraduate level, and within industry. In addition, two major areas of concern in software engineering education are discussed; namely, how to teach design and how to conduct software project courses. It is
apparent from these papers that a great deal is known about software engineering education. It is also apparent that a great deal more remains to be discovered in order for software engineering to become a mature intellectual discipline.

Richard E. Fairley
Guest Editor

REFERENCES


Richard E. Fairley received the B.S. degree in electrical engineering from the University of Missouri, the M.S. degree in electrical engineering from the University of New Mexico, Albuquerque, and the Ph.D. in computer science from the University of California, Los Angeles.

He is a Professor of Information Technology at George Mason University in Fairfax, VA, where he teaches courses and conducts research in the areas of software technology, project management, and other topics in software engineering. He was the Chairman and Chief Administrator of the School of Information Technology at Wang Institute of Graduate Studies, until the School was closed in August of 1987. He has worked in the computer field since the late 1950’s on a wide variety of machines and in a wide variety of programming languages. His work experience includes four years with the United States Air Force, four years with Sandia Corporation, two years with S&T Associates, and sixteen years as a teacher, researcher, and consultant. He has held tenured faculty positions at Texas A&M University, Colorado State University, and (currently) at George Mason University.

Dr. Fairley is Chairman of the Advisory Committee to the Software and Systems Resource Center at the Jet Propulsion Laboratory in Pasadena, CA; Co-chairman of a subcommittee of the IEEE Computer Society to develop a standard for software project management plans; and a member of the Advisory Committee to the Education Division of the Software Engineering Institute at Carnegie-Mellon University. He is Program Chair for the 11th International Conference on Software Engineering. He also provides consulting services and conducts short courses for leading companies in the computing industry. He is former executive editor of the Journal of Systems and Software, a former ACM National Lecturer, and he has edited three volumes published by Springer-Verlag: Software Engineering Tools (1980), Software Engineering Education (1986), and Educational Issues in Software Engineering (1987). He is also the author of the textbook Software Engineering Concepts (McGraw-Hill, 1985). In addition, he has published research papers on a wide variety of topics in computer science and software engineering. He is the author of three video tape series: Introduction to Software Engineering, The Ada Programming Language, and Software Project Management.