First Draft
12/78

MSE79: FIRST DRAFT OF A MASTERS CURRICULUM IN SOFTWARE ENGINEERING

Richard E. Fairley*
Computer Science Department
Colorado State University
Fort Collins, CO 80523
(303) 491-7026

Abstract

This paper is a status report on the curricular efforts of the IEEE Computer Society's subcommittee on software engineering education. The first draft of a proposed Masters program in software engineering is presented, as is the undergraduate preparation required for admission to the program. Potential implementation problems are discussed, and future plans are mentioned.

Introduction

Ten years have passed since the first conference on software engineering was held in Garmisch, Germany (1). At that time, the term "software engineering" was coined as a provocative title for a conference (two conferences) to consider the technological aspects of software development. During the intervening decade, software engineering has evolved into a major subdiscipline of computer science. Although much remains to be done, a body of knowledge and a set of guidelines have emerged which are concerned with improving the quality of software, and the techniques used to produce and maintain it.

Major milestones in the brief history of software engineering include the NATO Conferences mentioned above, the International Conferences on Software Engineering (2,3,4), introduction of the IEEE Transactions on Software Engineering(5), and the formation and rapid growth of the Computer Society's Technical Committee on Software Engineering and ACM's Special Interest Group on Software Engineering.

The demand for individuals trained in the skills of software engineering has become acute as computing systems have become more numerous, more complex, and more deeply embedded into modern society. Software engineering has been a growing concern to educators, and to perceptive individuals in industry, research labs, and government. Each of the three International Conferences on Software Engineering has had a session to consider educational issues, and several papers dealing with software engineering education have been published in other conferences and in the Transactions on Software Engineering. In 1975, IBM Canada sponsored a four day meeting to discuss software engineering education (6). In February, 1976, a one-day workshop titled "Software Engineering Education: Needs and Objectives" was held at the University of California, Irvine (7). Both meetings brought together educators and practitioners of software engineering to discuss issues in software engineering education. More recently, the status of software engineering education has been summarized in a paper by Wasserman and Freeman (8).

In recognition of the increasing concern for software engineering education, the Computer Society is sponsoring the development of a model curriculum in software engineering. This paper is the first draft of the proposed curriculum. It is not a formal recommendation or a final report; it is a working paper which is being published for comment and criticism.

---

*Chairman
IEEE Computer Society
Subcommittee on Software Engineering Education

First Draft
12/78

Background

The primary issue in software engineering education is the level at which software engineering skills should be taught. On the one hand, many individuals who will become practicing software engineers will be Bachelors graduates, and advocates of undergraduate software engineering education cite the success of electrical engineering educators in training electronic designers at the Bachelors level. On the other hand, software engineering is not (or perhaps has not yet evolved into) a discipline in which concepts and job assignments can be neatly compartmentalized. Instead, a software engineer is a generalist who is involved with applying computer science concepts and software engineering techniques to the analysis, design, implementation, validation, and maintenance of software systems. In order to accomplish these tasks, a software engineer must be technically competent, and in addition, highly skilled in the management and communications aspects of software engineering.

The majority of the subcommittee (although not unanimously) agree that the acquisition of software engineering skills requires maturity and motivation that can only be gained through experience with software. This maturity might be gained by undergraduate education, industrial experience, or by a combination of education and experience. The ideal entrant into a graduate program in software engineering would have an undergraduate degree in computer science, and two years experience with large software systems. Thus, the subcommittee (with one or two dissentions) recommends that software engineering be taught as a professional degree program at the Masters level.

An undergraduate "core" which provides minimum preparation for a Masters program in software engineeringis presented in Table I. Content of the core courses is specified by referencing the corresponding courses in the IEEE Model Curriculum and the new ACM Curriculum 78 (9,10). It is also assumed that undergraduate students will complete a mathematics sequence encompassing differential and integral calculus, linear algebra, and probability and statistics. In addition, undergraduate courses in accounting, technical writing, public speaking, and organizational behavior are highly desirable. Some of the material described in Table I may have been gained by the student through industrial experience, and it is expected that a Masters program in software engineering would be accessible to students with a mixed background of formal education and practical experience.

The undergraduate core has been designed for implementation in diverse educational environments; it is a starting point for those institutions desiring a program in software engineering. A student completing the undergraduate core will have been exposed to the fundamental concepts of computer science and computer engineering, which span the spectrum from digital logic to architecture to programming and software systems to compute science theory. The student will thus be prepared to pursue an in-depth program of study in software engineering; however, the undergraduate core does not, in itself, provide adequate training for a professional level software engineer. For instance, there are no courses in advanced programming methodology, or management issues in software engineering, or other topics that appear in the Masters program.

First Draft
12/78

The Masters Curriculum

The overall structure of the proposed Masters curriculum in software engineering is illustrated in Figure 1. A brief description of each course follows:

MSE-1: Introduction to Software Engineering
This course provides an overview of the field. The economics of software, technical aspects of programming methodology, the software lifecycle, software tools and techniques, managerial aspects of software engineering, and communication skills in software engineering are surveyed.

MSE-2: Requirements, Specifications, and Standards
This course provides an in-depth examination of requirements analysis, specification techniques, automated tools, and various formal mechanisms. Standards are also discussed.

MSE-3: Technical Communication
This course examines the role of communication skills in software engineering. Both oral and written skills are emphasized. The format and content of proposals, reports, user's manuals, and project documentation are discussed, as are the form and content of oral presentations of various types.

MSE-4: Organizational Behavior
This is a traditional course in organizational behavior, emphasizing the characteristics of technical organizations and creative individuals. Emphasis is placed on the need for management, on being managed, and on being a manager.

MSE-5: Software Design and Programming Methodology
This course emphasizes the techniques and practices used to transform a software specification into an operational system. It covers the various notations and methods available for organizing the design process, making and recording design decisions, evaluating designs, and transforming a design into executable code.

MSE-6: Security and Privacy
This course covers the technical aspects of data security: encryption techniques, data base security, and implementation of protection schemes in operating systems and programming languages. In addition, the legal and ethical aspects of security and privacy are emphasized.

MSE-7: Software Project Management
This course provides an in-depth examination of the management issues in software engineering: task organization, resource allocation, project milestones, product visibility, quality assurance, configuration control, cost estimation, forecasting, scheduling and budgetting, etc. In addition, the legal aspects of software engineering (contracts, liabilities, deliverable items, and acceptance criteria) are discussed, as are the ethics of software engineering.

MSE-8: Software Laboratory
This course is a two semester lab sequence in which students work in teams to specify, design, implement, test, document, and modify a large software system, using tools and techniques discussed in other courses.

First Draft
12/78

MSE-9: Validation, Verification, and Performance Measurement
This course discusses validation and verification techniques such as proofs of correctness, structured walkthroughs, static analysis, dynamic testing, symbolic execution, and continuous verification. Reliability models for software are also discussed. In addition performance measurement techniques, analytical models, simulation models, and statistical methods are discussed.

MSE-10: Human Factors in Computing System Design
This course examines automation of user processes, design of user interfaces, data presentation techniques, and the human factors aspects of operations and maintenance procedures. In addition, human factors in the software development process are discussed.

MSE-11: Data Base Systems
This course considers data base systems from several different views: The use of a data base system to automate various aspects of the software development process, human engineering of data base systems, and the design and implementation of data base systems using the techniques of software engineering.

MSE-12: Distributed Computing Systems
This course deals with the design and analysis of architecture and software for the full spectrum of distributed systems, ranging from loosely coupled networks to tightly coupled multiprocessors. In addition, techniques for networking mini and micro computers are considered.

Each of these courses is designed as a three semester credit hour course. In addition to the year long lab sequence, it is assumed that each course will use homework and laboratory assignments to reinforce the material.

Implementation Problems

Many problems will arise in the implementation of a software engineering program, not the least of which are the political problems (11). These include the name itself, "software engineering", determining which department in the university will have jurisdiction over the program, and determining whether the program will be an undergraduate program, a gradute program, or a combination.

There are at least three departments in most universities that have a legitimate interest in teaching software engineering courses: the computer science department, the electrical engineering department, and the management information systems department. It is unfortunate that the name "software engineering" implies implementation of the program by an engineering department. The curriculum has been designed without bias toward any particular department in the university, and we foresee that this body of material will be taught under many different names, in many different departments.

Other problems that will arise in the implementation of software engineering programs include the lack of qualified faculty, lack of adequate textual materials, development of suitable laboratory problems, and providing realism in an educational environment. These, and other issues are discussed in the references cited (8,11).

First Draft 12/78

Summary

This paper has presented the first draft of a proposed Masters program in software engineering. It is a report on work in progress and should be read with that in mind. Perhaps it is premature to develop extensive curricula in a young and rapidly evolving field; yet, the need is obvious. Our goals in preparing the recommendation are to provide a sharper definition of the field, to indicate the educational materials that must be developed to support the proposed program of study, and to provide guidance for the implementation of software engineering programs.

Currently, a rough draft of the detailed course outlines is being distributed to a review committee of 50 to 60 well known individuals in software engineering. Their recommendations will be incorporated into a final draft, which will be published in spring, 1979. The total curriculum will exist in only a few institutions, if at all. Some schools will implement subsets of the proposed curriculum, and it will be modified and extended in various ways according to the needs of local students and local industries, local credit hour and budgetary restrictions, local faculty talents, and other constraints. If this curriculum can serve a wide variety of institutions as a reference point, and a starting point for curriculum design in software engineering, then we will have achieved our purpose. We welcome your comments and criticisms of this proposal, as well as your opinions of the issues discussed in the paper.

Acknowledgements

The following individuals designed the proposed curriculum and developed the detailed course outlines: Bruce Barnes - NSF, George Davida - UW Milwaukee, Randy Jensen - Hughes Aircraft, Hisashi Kobayashi - IBM, Keith McCammon - DEC, Mike Mulder - Dept. of Energy, Dave Rine - Western Illinois Univ., Leon Stucki - Boeing Computer Services, Tony Wasserman - UCSF, and Andy Tanenbaum - Free Univ. Amsterdam.

References

1. P. Naur, B. Randell, and J. Buxton, Software Engineering: Concepts and Techniques. Petrocelli/Charter, New York, 1976.
2. Conference Proceedings, 1st National Conference on Software Engineering, IEEE Catalog No. 75CH0992-8C, 1975.
3. Conference Proceedings, 2nd International Conference on Software Engineering, IEEE Catalog No. 76CH1125-4C, 1976.
4. Conference Proceedings, 3rd International Conference on Software Engineering, IEEE Catalog No. 78CH1317-7C, 1978.
5. Bimonthly Journal, IEEE Transactions on Software Engineering, Published by the IEEE Computer Society.
6. D. Oakes, et al, Eds.,Software Engineering Education: Proceedings IBM Scientific Symposium, IBM Canada (available from INFOR Journal, Ottawa, Ontario)
7. P. Freeman and A. Wasserman, Eds., Software Engineering Education: Needs and Objectives, Springer-Verlag, New York, 1976.
8. A. Wasserman and P. Freeman, "Software Engineering Education: Status and Prospects", Proc of the IEEE, Vol.66, No.8, August, 1978.
9. Committee Report, A Curriculum in Computer Science and Engineering, IEEE Catalog No. EH0119-8, January, 1977.
10. Committee Report, Curriculum Recommendations for the Undergraduate Program in Computer Science, A Report of the ACM Curriculum Committee on Computer Science, to appear in the Comm. of the ACM.
11. R. Fairley, "Educational Issues in Software Engineering", Proc. of the 1978 ACM National Conference, Washington, D.C., December, 1978.

First Draft
12/78 Table I. The Undergraduate Prerequisite Program

CORE COURSE	COMPUTER SOCIETY REFERENCE COURSE(S)	ACM 78 REFERENCE COURSE(S)
Digital Logic	DL-1; DL-2	CS 4
Digital Lab	L-1	
Computer Organization	CO-1; CO-2; CO-3	CS 3; CS 4
Microprocessors	DL-3	
Microprocessor Lab	L-3	
Introduction to Computer Programming	SE-1	CS 1; CS 2
Data Structures and Design of Algorithms	SE-2; SE-3	CS 7
Operating Systems and Computer Architect.	SE-6; SE-7	CS 6
Data Base Systems	SE-4	CS 11
Survey of Language Concepts	SE-5	CS 8
Language Implementation	SE-8	CS 15
Discrete Structures	TC-1	MA 4
Design and Analysis of Algorithms	TC-2	CS 13
Software Development Tools and Techniques		

FIRST SEMESTER	MSE-1 Introduction to Software Engineering	MSE-2 Requirements, Specifications, and Standards	MSE-3 Technical Communication	MSE-4 Organizational Behavior
SECOND SEMESTER		MSE-5 Software Design and Programming Methodology	MSE-6 Security and Privacy	MSE-7 Software Project Management
THIRD SEMESTER	MSE-8 Software Laboratory Lab 1	MSE-9 Validation, Verification, and Performance Measurement		MSE-10 Human Factors in Computing System Design
FOURTH SEMESTER	MSE-8 Software Laboratory Lab 2		MSE-11 Data Base Systems	MSE-12 Distributed Systems

Figure 1.- A Proposed Masters Program in Software Engineering