MASTER OF SOFTWARE ENGINEERING - A PROPOSED CURRICULUM FOR PRACTITIONERS

A. A. J. Hoffman
Texas Christian University
Fort Worth, TX 76129

It is well known that software development projects usually exceed both time and budget projections. Furthermore, many software systems do not meet expected performance, are difficult to modify and maintain, and often have unexpectedly short life cycles. These problems exist even when the developers are college graduates with bachelor, masters, and doctoral degrees in computer science. Clearly, the problem is not lack of training but rather focuses on the need for a new approach to the software development process.

"Software Engineering" is a term coined in 1967 by a computer science study group of the NATO Science Committee as the theme of a workshop on improving the process of software development.

The purpose of the NATO workshop was to attempt to transform software design and development into an engineering-type discipline. Up to very recently, the task has involved seeking out and using techniques which can assist in the economic development of software which executes reliably and efficiently on real machines. This collection of ad hoc techniques is correctly named "software design methodologies" and often incorrectly referred to as "software engineering." Software design methodologies (also called improved programming methodologies) are enormously helpful but do not in themselves solve the basic problems of software development. Now software engineering refers to the application of the principles of applied computer science (which includes software design methodologies), management science, and communication skills to the economic design, development, implementation, and maintenance of software systems. Programming involves simply converting a given set of specifications into computer executable code. Solving the basic software system development problems involves dealing with the entire software life cycle which spans the time from conception of the product to the end of its operational life. Software projects typically involve a myriad of managers, engineers, accountants, programmers, and customers - a fact which leads to the need for application of the best available technologies in an environment where there is a need for effective management and communications.

The overall task of software development involves at least the following areas:

1. systematic programming methodologies,
2. requirements analysis,
3. writing specifications,
4. design technology,
5. coding,
6. testing, validation, verification certification, security,
7. management,
8. economics,
9. group dynamics,
10. oral/visual communication,
11. modification, maintenance, portability.

These areas or tasks can be conveniently grouped into three skill areas:

A. Systematic Programming Methodologies,
B. Management, and
C. Communication.

It is granted that, in all of these areas, there is no substitute for maturity and experience, However, few practitioners (i.e. software developers) currently in the field have educational background in all three areas. Here an attempt is made to address the problem of introducing these skills into the software development community through the Master of Software Engineering.

In considering the types of individuals who might be interested in a Masters degree program in software engineering are:

A. Persons who have completed a Bachelors Degree in Computer Science or a related field with little or no software development experience,

B. Persons with substantial job related experience but either no formal education in any of the three areas or no recent formal education in computer science.

Individuals with no experience in software development have difficulty in perceiving the intrinsic differences between one-person and multi-person projects and the associated problems. Persons with extensive experience already "know how to write code" but may not understand why systems tend to exceed both budget and time projections, don't work as well as expected, and are difficult to modify and maintain.

This Master of Software Engineering curriculum is designed to serve this constituency and bring them to the point of productive participants on highly successful software projects or managers of software development projects.

To enter the program the prospective student should gain admission to Graduate School and have either

1. B.S. in Computer Science and some software development experience,

or

2. Substantial job related experience in software development.

The program should provide an introductory course to motivate the student and provide an overview of the problems of the software development process and a description of the software engineering tools (or skills) available to address these problems. In addition, the program should contain an optional course (or courses) for those lacking a formal or recent computer science technical educational background. As already mentioned, management and communication courses should complement the technology courses. The structure and flow of courses is shown in Figure 1. To serve these purposes a set of eleven sample courses are presented:

SE-1: Introduction to Software Engineering
(3 hours)

Deals with the following questions: What are the problems encountered in software development? What are the intrinsic differences between one-person efforts and multi-person software projects? What tools are available to deal with these problems? What is the state of the art?

SE-2: Overview of Computer Science
(3 or 4 hours)

Overview of undergraduate computer science from a software engineering point of view. Required only for those lacking a formal or recent computer science background.

SE-3: Methodologies
(3 hours)

Structured programming. Modularization. Top-down development. Levels of abstraction. Stepwise refinement. Hardware, software, and user trade-offs.

SE-4: Requirements and Specifications
(2 hours lecture + 1 hour laboratory)

Requirements analysis. Techniques for representing requirements. Specification development techniques. Specification languages. Automated aids. Laboratory will consist of case studies.

SE-5: Design
(3 hours with 2 hour parallel laboratory...SE-6)

The design process. Major design methods such as composite/structured design, data structure driven design, structural analysis, and others. Evaluation of alternate designs. Automated design aids. Design documentation.

SE-6: Design Laboratory
(2 hours - parallel to SE-5)

Case study designs using design methods contained in SE-5.

SE-7: Implementation
(3 hours)

Transfer of design to code. Testing techniques. Validation. Verification. Certification. Security. Case studies.

SE-8: Management of Software Development
(3 hours)

Organization context of software development. Analysis of life cycle costs. Scheduling and budgeting techniques. Specification and control of standards for products, processes, and equipment. Personnel development and utilization. Team techniques.

SE-9: Economics of Software Development
(3 hours)

Fundamentals of economics. Distribution of costs through software life cycle. Relative hardware/software costs. Economic analysis for decision-making. Economic feasibility studies.

SE-10: Effective Participation in Small Task Oriented Groups
(3 hours)

Recognizing and supplying actions necessary to achieve their objectives. Group maintenance roles. Group orienting roles. Task directed roles. Evaluative roles. Closure and action items. Systematic approaches to problem solving. Problem definition. Developing the solution domain. Means-end analysis. Pro-

visions for feedback. Delineation of subproblems. Assignment of priorities. Time lines. (Should be taken early in the program.)

SE-11: Communication Techniques for Software Engineers
(3 hours)

Organization of presentation materials. Preparation of graphics for presentation. Maximizing the use of multimedia. Writing style for software documents. Development documents - requirements, specifications, design, and implementation. Technical documentation. User documentation, automated aids. Reports. Proposals. (Should be taken after SE-3)

These courses follow the pattern shown in Figure 2. This plan would provide for a 36 semester hour Masters degree program with either

1. 32 hours of software engineering plus 4 hours of approved electives,

or

2. 29 hours of software engineering (without SE-2) plus 7 hours of electives.

Local options could include a thesis, a comprehensive examination, or a project. The student exiting from this program would have:

1. The ability to function in a senior level position in software development,
2. Could pursue further work in software engineering education (i.e. take courses to keep-up),
3. Read current application oriented literature, and
4. Could make intelligent choices in the development/design process.

Students who participated in the program on a part-time basis would expect to complete the degree in approximately three years.

An institution offering this program would have to have a faculty with extensive application oriented experience in the designated areas.

A. Persons who have completed a Bachelors Degree in Computer Science or a related field with little or no software development experience,

B. Persons with substantial job related experience but either no formal education in any of the three areas or no recent formal education in computer science.

Individuals with no experience in software development have difficulty in perceiving the intrinsic differences between one-person and multi-person projects and the associated problems. Persons with extensive experience already "know how to write code" but may not understand why systems tend to exceed both budget and time projections, don't work as well as expected, and are difficult to modify and maintain.

This Master of Software Engineering curriculum is designed to serve this constituency and bring them to the point of productive participants on highly successful software projects or managers of software development projects.

To enter the program the prospective student should gain admission to Graduate School and have either

1. B.S. in Computer Science and some software development experience,

or

2. Substantial job related experi- experience in software development.

The program should provide an introductory course to mativate the student and provide an overview of the problems of the software development process and a description of the software engineering tools

MASTER OF SOFTWARE ENGINEERING

Subjects - Sequence

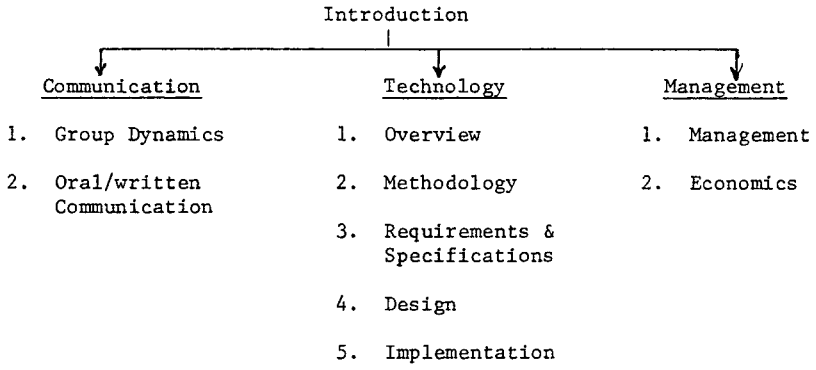

Figure 1

MASTER OF SOFTWARE ENGINEERING

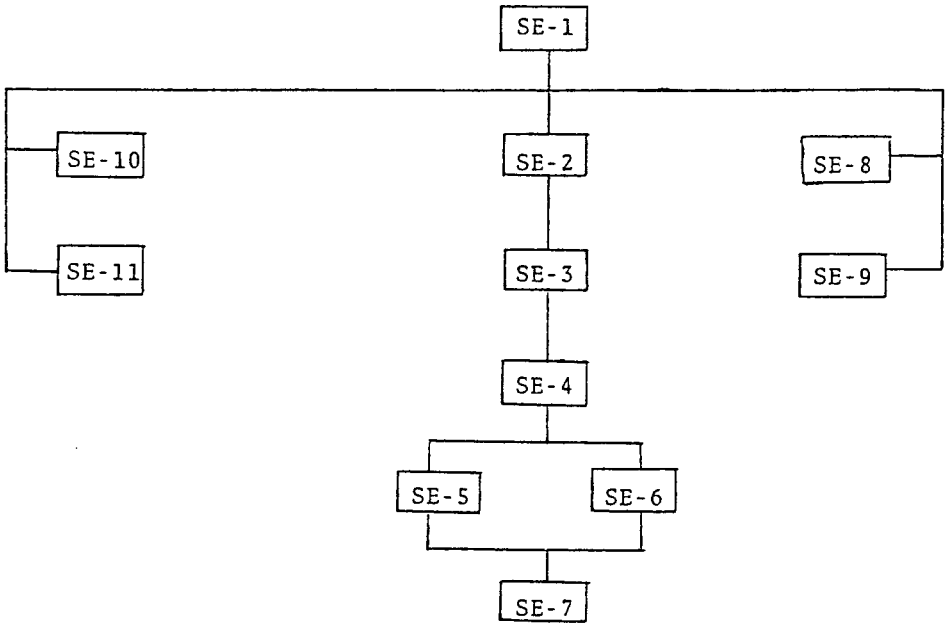

Figure 2