The Master of Software Engineering Degree: An Integrative Engineering Discipline

David Gustafson, Bill Hankley, and Virg Wallentine
Department of Computing and Information Sciences,
Kansas State University, Manhattan, KS 66506
Tel: (913)532-6350

August 31, 1995

Abstract

Software-based products are an ever-increasing portion of industrial output. However, few engineers who produce software have had formal training in the specification, design, implementation, documentation, and maintenance of large software systems. In addition, few have had training in software engineering management and development tools and processes. KSU has implemented a Master of Software Engineering degree program which provides an opportunity for new engineering graduates and practicing engineers to learn state-of-the-art software engineering analysis and processes. The purpose is to provide students a foundation in the software engineering "life cycle", software measurement, software management, software specification, and software validation and verification. In addition to the core courses on software methodology, students are required to take courses in an application area which is reliant on software development. This involves faculty from many engineering and science disciplines who teach these courses and supervise the development of the student's software "portfolio" in the designated engineering or science area. The goal is to integrate the software engineering process into traditional engineering processes.

1 Introduction

In "Software's Chronic Crisis, W. Wayt Gibbs [1] quotes Remi H. Bourgonjon, director of software technology at Philips Research Laboratory in Eindhoven as saying, "The amount of code in most consumer products is doubling every two years." Much of the software developed today has been produced by engineers and scientists who have no formal training in software engineering. Industry is replete with examples of poorly constructed software which does not meet deadlines, accuracy, and cost constraints. Gibbs also writes, "A quarter of a century later software engineering remains a term of aspiration. The vast majority of computer code is still handcrafted from raw programming languges by artisans using techniques they neither measure nor are able to repeat consistently." He further states "Disaster will become an increasingly common and disruptive part of software development unless programming takes on more of the characteristics of an engineering discipline rooted firmly in science and mathematics."

2 The Software Engineering Discipline

The discipline of software engineering covers the application of engineering principles to the building of computer software. The field covers the theories, tools and methods for systematic representation, design, verification, development, production, validation, and maintenance of software products including programs, prototypes, documentation, user interfaces, training, and evaluation. Software engineering is applicable not only to computer systems software; the techniques of software engineering offer benefit for software developed for all disciplines.

Engineering is the "practical application of scientific knowledge." The application of knowledge about software is made practical through the use of common techniques, components, tools, and methods of management. Specifically, like other branches of engineering, software engineering:

- uses formal models and methods of computing to develop formal requirements of and specifications for application domain software;
- uses established techniques of design to establish structure of software before it is programmed;

- uses established techniques for verification (formal analysis of correctness of properties) of system design;
- uses established techniques for validation (systematic measurement and analysis of properties) of systems implementation;
- emphasizes the building of large software systems by integration of standardized components;
- uses numerous software tools to provide assistance in all activities of software development;
- studies the processes of software development as the basis for systematic management.

3 Computer Science vs. Software Engineering

There is much discussion about the relationship between software engineering and computer science. One view is that software engineering is just a subpart of computer science. This is much like the view that electrical theory is a part of physics. This view says that the theoretical aspects of software engineering are those of computer science and that software development is the simple application of computer science theory.

The other view is that software engineering is to computer science what electrical engineering is to physics. This view holds that software engineering is an independent area of study with its own theoretical components about the development of software. Many proponents of this view recommend a separate department for software engineering.

Although our master of software engineering program is part of the computing and information science department, we recognize that it is a separate area of study. The concerns of software engineering are not identical to those of traditional computer science. There are distinctly different areas of concern and a standard master's program in computer science will not satisfy all of the needs of software developers.

4 Integrating SE into Traditional Engineering Processes

In "Computing the Future: A Broader Agenda for Computer Science and Engineering", the National Research Council [2] recommends that we "Look outward as well as inward. A broader agenda would legitimize closer couplings to science, engineering, commerce, and industry." If we are to be successful in reversing this trend of poorly constructed software, we must train both new engineers and practicing engineers in the practice of software engineering.

5 The MSE Curriculum

An MSE curriculum should contain the kinds of courses that prepare the student to understand software processes, to manage these processes effectively, and to apply these processes to their own specific area of expertise. Our goal in the MSE program is to provide state-of-the-art software engineering technology to both the practicing engineer and to the new science and engineering graduate. It is our view that a graduate from an MSE program should have fundamental computing expertise, a solid grounding in the software "lifecycle", exposure to software development tools, knowledge of measurement and management of software development, and extensive experience in applying software engineering methods to their specific discipline.

An illustration of the courses in the KSU MSE program is given in Figure 1. Students are assumed to have some computer science background, including logic, programming, data structures, and algorithms. They must also have some coursework or professional experience in software engineering. The students are also required to have some background in their own discipline, such as computer science or engineering. The first two required courses in the MSE expose the students to the software engineering life cycle and to formal specification of programs. Then, the student must take two second level software engineering courses. At KSU the students can pick any two of the following courses: object-oriented programming, advanced computer networks, data base design, management of software development, and software metrics. We feel that formal models of both distributed systems protocol design and data base design methods are appropriate for software engineers, as well as traditional software engineering methods. At the advanced level, we

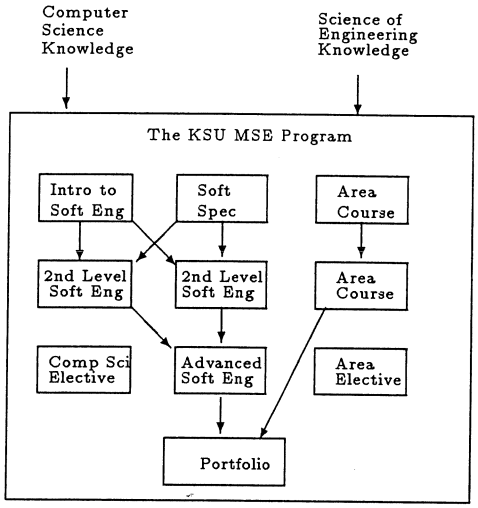

Figure 1: The KSU Masters of Software Engineering Program

require students to take one of the following four courses: protocol engineering, software validation, specification and verification of reactive systems, or data engineering. Finally, since we want traditional engineering and science students in the MSE, we require them to take two courses in their own area, such as power engineering, computer engineering, robotics, data systems, distributed systems, etc. At this point, students should have the proper background to develop a project in their own area of expertise, using the current state-of-the-art software engineering methods. This project, which will result in their software portfolio, will be under the direction of both a software engineer and a discipline expert.

An integral part of preparing engineering professionals is to give them experience in design. The software portfolio is a concept in the KSU MSE degree program which attempts to apply this concept to software engineering. Just as someone in marketing or art would bring a portfolio of their work to a prospective client or employer, we believe the software portfolio should represent the types of projects a software engineer has produced. Thus, the overall goal of this (significant) part of an MSE degree program is for the student to produce a set of software artifacts which demonstrate a student's ability in all phases and activities of software development. The contents of the portfolio are examples of those phases and activities. The quality of these examples should be high. The professionalism of the student should be

evident from these examples. The contents should include an overview, a cost estimate, a project plan, a requirements specification, a formal requirements section, a test plan, a software quality assurance plan, a design, the source code, the testing summary, and a reliability summary. In the KSU MSE program, these attributes of good projects can be demonstrated in one large software system or in a set of smaller, but complete, projects.

Finally, two courses are electives and can be taken either in computer science or in their own area of expertise. A more detailed description of the KSU MSE progam is contained in the Appendix.

The MSE curriculum is, by design, intended to stimulate the integration of software engineering technology and methods with traditional engineering disciplines. Thus, there is a necessity for involving both software engineering faculty and traditional engineering faculty in the design of the student's program of study, in the teaching of the courses, and in the construction of the software portfolio. This may sound very sensible and easy-to-do, but when two different cultures come together, sometimes the philosophies collide and sometimes they mesh. Thus, one of the most important aspects of this program is to involve both engineering cultures early in the process. It pays dividends in many ways, including joint teaching and joint research and development projects.

6 The Delivery

In keeping with the goal of reaching both new science and engineering graduates and working professionals, several different kinds of delivery mechanisms must be utilized. At KSU, we deliver the MSE in three ways. First, we offer courses on-campus. Second, we offer a summer-on-campus program. And, third, we offer the MSE courses via both live video and videotape delivery. The on-campus courses are held in TV classrooms and these are both videotaped and broadcast live to various industries, which have audio feedback for student-faculty interaction. The remainder of the program is carried-out through electronic mail, telephone, fax, and our KSU home page on the World Wide Webb. The summer-on-campus program is for companies who can afford to send their employees to KSU for five weeks each summer. By also taking one course per semester through video, these students can get their degree in three summers.

7 Conclusion

Although the MSE program has just recently been approved by the Kansas Board of Regents, the degree program is quite popular and we will graduate our first students in December, 1995. Our current admissions indicate that about half the MSE students will be from our summer-on-campus program, about one third from other off-campus students and about one sixth from the traditional on-campus students. We feel that our program fulfills a need that we had identified among our potential students and we are excited about serving those students.

8 References

1. Gibbs, W.W. Software's Chronic Crisis. Scientific American, September, 1994.

2. National Academy of Sciences. Computing the Future: A Broader Agenda for Computer Science and Engineering. National Academy Press, Washington, D.C., 1992.

3. Ford. 1991 SEI Report on Graduate Software Engineering Education. Tech. Rpt. CMU/SEI-91-TR-2. Software Engineering Institute. Carnegie-Mellon University, April, 1991.

A The KSU MSE Program

A.1 Admission to the MSE Program

The admission requirements for the proposed MSE program are:

1. a baccalaureate degree in computer science, computer engineering, a mathematical science, or a related engineering or science area and

2. (undergraduate) courses in programming, data structures, and algorithms and software engineering or equivalent practical software engineering experience.

A.2 Program of Study

The Program of Study in the proposed MSE program will consist of 33 semester credits that must include the following:

1. CIS 740 and CIS 771
2. two courses from CIS 644, CIS 725, CIS 746, CIS 748, and CIS 764.
3. one course from CIS 826, CIS 841, CIS 842, and CIS 864.
4. two courses from an application area such as: Parallel and Distributed Systems, Operating Systems, Realtime Systems, Database Engineering, Knowledge-based Systems, Graphics, or specialty areas from other enginering areas.
5. CIS 895 MSE Project (six credits)
6. six credits of technical electives (computer science or application area courses).

A.3 Application Area

Each student will specialize in an application area. It is expected that each student will do his/her project in an area related to that application area and that one supervisory committee member will have expertise in the application area. Each student will produce a "software portfolio" which will contain a representation of the student's most important software expertise. The electives will allow the MSE students the opportunity to strengthentheir overall computer science skills or to gain some expertise in related areas.

A.4 Core Courses

- CIS 644 Object Oriented Software Development
- CIS 725 Advanced Computer Networks
- CIS 740 Software Engineering
- CIS 746 Software Measurement
- CIS 748 Software Management

- CIS 764 Database Design
- CIS 771 Software Specification
- CIS 826 Protocol Engineering
- CIS 841 Software Validation
- CIS 842 Specification and Verification of Reactive Systems
- CIS 864 Data Engineering
- CIS 895 MSE Project