Proposal for a B.S. Degree in Software Engineering

Michigan Technological University
Department of Computer Science

March 11, 2003

1 Executive Summary

It is becoming apparent that software developers with a solid understanding of engineering principles produce high quality software in an efficient manner. Employers are beginning to put a premium on graduates with knowledge of software engineering, particularly those with knowledge in particular application areas. Yet undergraduate degree programs in software engineering are still rare in the United States. We plan to establish a B.S. degree in software engineering that takes advantage of this university's well-established Computer Science program, as well as its wealth of expertise in engineering, science, and business. Students in this degree program will study an application area of their choosing. Expertise in these areas will come both from faculty in other departments and from industrial partners. Students will also learn about the business side of software development from experts in management and entrepreneurship. We believe that the practical, interdisciplinary nature of this program will help attract students and retain them once they are here.

2 Need for Proposed Program

The mission statement of the university's Strategic Plan reads, "We prepare students to create the future". It is clear that whatever form the future takes, computers will be used to create it. We must ensure that we are providing degree options that prepare our students for what they will find in their professional careers. An alumnus recently remarked in a conversation with our students that computing professions are becoming more highly specialized. It is increasingly rare for a single computer scientist to be hired to do a wide range of activities. Instead, there are network engineers, system administrators, software developers, database administrators, software engineers and computer engineers, to name just a few of the many types of computing positions currently available. As career options become more highly focused, educational programs need to support greater specialization while still providing the breadth necessary to remain flexible as career opportunities evolve.

Software engineering has emerged as a response to the increasingly specialized and applied uses of computers. A leading figure in the area [13] describes this new field in the following way. Software engineers must be able to use the body of knowledge and methods created by computer scientists; they must also be able to apply broader areas of knowledge in particular application domains. They use this knowledge within a well-defined discipline of design and analysis that produces real software products. A recent scan of the Michigan Tech Computer Science alumni list reveals that many are in positions that involve software engineering: "software engineer", "software architect", "software test engineer", "development engineer", "senior design engineer", "quality assurance engineer". This appears to be part of a national trend: the U.S. Department of Labor projects that software engineering will be the fastest-growing occupation over the 2000–2010 period [1].

As a first step toward providing this more focused education, the Department of Computer Science established a Software Engineering option within the B.S. Degree in Computer Science two years ago. The next step is a B.S. Degree in Software Engineering. The proposed program is designed to take advantage of the University's engineering, science and business expertise. Students will develop domain knowledge in an application area of their choosing. Furthermore, since many Software Engineering graduates will ultimately find themselves in project management positions, we plan to prepare them by including a management component. Another goal is to provide students interested in someday starting their own software-related business with an option to develop expertise in entrepreneurship.

3 Relevance to University Strategic Plan

The new Software Engineering degree program will provide the following benefits in relationship to the University goals:

- **Attract and retain high quality students.** Our experience indicates that ever more high school students are thinking of careers in software engineering. A distinct software engineering degree program will surely be of interest to these students. This supports Goal 3 "Size and Composition" of the Strategic Plan.

- **Increase national visibility.** Currently only a few major research universities have degree programs in software engineering. Being an early provider of such a degree program will enhance our national visibility. This supports Goal 7 "Image".

- **Enhance the education of students in other disciplines.** In this degree program, students will develop expertise in an application area through coursework and project experience. Through the project, Software Engineering students will work with other students who have expertise in the application area but know little about software engineering. Since many of these other students will ultimately use or develop products that contain software components, they will be better prepared by observing how software can be engineered. Furthermore, many senior design projects and enterprises

in the College of Engineering have a significant software component. We plan to encourage Software Engineering students to participate in these projects and enterprises. This supports Goal 1 "Learning".

- **Increase interdisciplinary scholarship.** This degree program will increase interaction between Software Engineering faculty and faculty in other areas. A beneficial consequence will be heightened awareness of potential interdisciplinary research problems. In many cases, the interdisciplinary interests may go beyond software engineering and involve Computer Science faculty with expertise in areas such as artificial intelligence, graphics, or computational science. This supports Goal 2 "Scholarship".

- **Support outreach.** A software engineering degree will enhance our interactions with industry, both locally and nationally. In particular, student projects may be obtained from local companies established through the Michigan Technological University Enterprise Smartzone. Other projects may lead to commercialization or the establishment of local companies. This supports Goal 6 "Outreach".

A new Software Engineering degree program should significantly aid in the University's recruitment of high quality students. We have noticed two trends that indicate a high potential demand for such a program. First, ever-increasing numbers of prospective students visiting campus express interest in software engineering. Soon such students will be looking specifically for a software engineering degree program, not a computer science degree program. Second, the enrollment in our Software Engineering senior level course for Spring 2002 was more than double the Spring 2001 enrollment, which was prior to the announcement of the Software Engineering option. Software Engineering, along with the existing programs in Computer Science, Computer Engineering, the Management Information Systems option in Business & Economics, and the proposed Computer Systems Science program, should form an excellent mix of computer-related degree programs. We will be able to offer high school students a package of exciting marketing material, presenting a wide array of educational opportunities in the computing field. The Department has made a particular effort to design our three degree programs (computer science, computer systems science, and software engineering) with sufficient commonality the first two years to allow students to easily switch between these majors. This should provide them with more time to make an informed choice concerning the appropriate degree program for their desired career. This diversity of degree options is likely to increase student retention as students become more aware of the diversity of career opportunities available to them.

Nationally, the enrollment of women in computing fields is alarmingly low, and this University is no exception. In Fall 2002, only 8.5% of the students working on a degree in Computer Science at Michigan Tech and 12% of the students working on a degree in Computer Engineering were women. The Computer Science Department at Carnegie-Mellon University has conducted extensive studies on why women are so underrepresented in computing fields. Margolis *et al.* have found that women tend to be more interested in "computing for a purpose" [6, 7]. That is, women place a higher value on the context of computing and its ties to other fields. We believe that the new Software Engineering degree program, which

explicitly connects computer science to real-world problems via the domain knowledge requirement, will appeal more to both women and men who are interested in the wider context of computing.

Independently of this proposal, the Computer Science Department is conducting an investigation of retention issues. Much of what the Department learns about Computer Science student retention will also help us to understand how better to retain Software Engineering students. We are speculating that Software Engineering students will be easier to retain than Computer Science students. A surprising number of Computer Science students leave the discipline because they find they do not like programming and are concerned about doing it for the rest of their lives. Many of these students who leave computer science had minimal prior awareness of what the discipline of computer science entails. Presumably, Software Engineering students will enter the major with a greater awareness that programming is a major part of software development; we hope that they will finish the major aware that it is just one part of a multifaceted and exciting process.

4 Related Programs

Undergraduate degrees in Software Engineering are common in Canada, Europe, and most notably in Australia [3]. They are still rare in the United States, but the numbers are beginning to grow. The August 2002 issue of the Forum for Advancing Software Engineering Education (FASE), Volume 12 Number 08 (Issue 151), lists 17 schools in the U.S. that currently offer undergraduate degrees in software engineering. The earliest programs were implemented at Rochester Institute of Technology [15], Milwaukee School of Engineering [10], Mississippi State University [11], and Gannon University [2]. This year, both Rochester Institute of Technology and Milwaukee School of Engineering are seeking ABET accreditation for their software engineering programs. Within the State of Michigan, only University of Michigan, Dearborn offers a software engineering degree program [17]. The notion of a domain-specific software engineering curriculum with industrial ties was first explored at Embry-Riddle Aeronautical University and has since been embraced by institutions in Europe [8] and Australia [4], as well as Rochester Institute of Technology and the Milwaukee School of Engineering [16]. The reports from these pioneers has been universally and enthusiastically positive. We summarize some of the responses below.

Rochester Institute of Technology was the first American university to offer a bachelor's degree in software engineering [12]. Lutz [16] reports that the program proved successful even before the graduation of the first Software Engineering students, based on the near-100% student co-op placement. While RIT's enrollment in Computer Science grew at roughly the same impressive rate as the national average, the enrollment increase in Software Engineering outstripped even that of Computer Science. An "Industrial Advisory Committee" was formed to provide input to improve the program; as it has turned out, this committee has also become a major booster of the program. Lutz states that the decision to form the degree program "is already paying handsome dividends in prestige, recruitment, and enrollment" [16]. RIT requires students to complete three courses in an application domain. Example

application domains include manufacturing engineering, electrical engineering, mechanical engineering, scientific and engineering computing, commercial and management applications, and computer graphics.

The "Software Center" at Embry-Riddle Aeronautical University facilitates collaboration with industry [5]. The projects undertaken by the students reflect Embry-Riddle's strength in the domain of air transportation; examples include software for air-traffic control training and flight simulation. The program has attracted industrial giants like Lockheed-Martin. Kornecki *et al.* report that student satisfaction, as measured by lower attrition, higher enrollment, and higher job placement, has soared since Embry-Riddle adopted this approach.

Grant [3] reports that Software Engineering degree programs in Australia are extremely popular, despite the fact that they require an extra year of study and despite the lure of plentiful, easily obtainable employment opportunities. He finds that employers now recognize the benefits of solid engineering principles in software development and seek out students who have absorbed them in their education. Harrison [4], reporting on efforts at the University of Queensland in Australia, claims that students are more satisfied with their education as a result of ties with industry; putting the principles they have learned into practice confirms the value of what they have learned and provides context for what they learn later.

Our proposed software engineering program is similar to the one proposed by the University of Michigan, Dearborn. An important difference is that students in our program will be required to complete a sequence of courses in an application area. We are stressing the importance of acquiring domain knowledge.

5 Program Administration

Much of the organizational structure needed for this program is already in place in the Computer Science Department. The Computer Science curriculum that will form a basis for Software Engineering students' education is well defined, as are the undergraduate Software Engineering courses. Computer support staff, secretarial staff and faculty advisors already help Computer Science students. Industrial placement is done through the Career Center. However, the program will require a greater amount of advising, a more aggressive approach to forging links with industry and other academic departments, and a higher level of communication between the Department and industrial sponsors.

Each senior design project or enterprise will be assigned a faculty advisor, who will guide its progress and assess it at the end. Initially, Prof. Wallace will advise Software Engineering students in choosing appropriate courses in their desired application domains. Prof. Ott will work on attracting interest in industry and in other university departments.

A part-time *Industrial Project Coordinator* will serve as an intermediary between the project teams and the industrial partners. It will be the responsibility of the Coordinator to ensure timely completion of projects and good communication between the student teams and sponsors. This person will also be responsible for establishing the policies and procedures that will be necessary to conduct these projects with industry.

The program will be housed in the Department of Computer Science. A faculty com-

mittee (the *Software Engineering Program Committee*) will be responsible for the Software Engineering curriculum. This committee will work through the Department's Undergraduate Committee on all curricular issues. Course and curriculum proposals from the Software Engineering Program Committee go to the Undergraduate Committee for recommendations, which are then forwarded to the faculty. The Undergraduate Committee is responsible for ensuring that courses common to more than one degree serve all programs effectively. The Undergraduate Committee is also responsible for ensuring an appropriate balance in curricular issues across the different degrees offered by the Department. Finally, the application electives proposed by each Software Engineering student must be approved by the Undergraduate Committee, to ensure quality and consistency.

The Software Engineering Program Committee is responsible for consulting closely with project partners and other experts to ensure that the program remains relevant. A group of representatives of the various project sponsors will be asked to give feedback on project progress. In addition, an *Industrial Advisory Board* will be consulted for ways to improve the program. Board members will include Computer Science alumni and representatives from industries with connections to the Department.

6 Faculty Resources

The faculty in the Computer Science Department at Michigan Technological University have been very successful in preparing students for careers in software engineering, as demonstrated by the large number of alumni employed as software engineers. Examples of the diversity of application areas in which alums are working include computer games, digital signal processors, chemical manufacturing, biomedical applications, telecommunications, web-based applications, software development tools, data management applications, aeronautics, aerospace, insurance, embedded automotive applications, food production, factory control, embedded control devices, robotics, bioinformatics, communications systems, sensors, health care, film production, enterprise software, pharmaceuticals, computer hardware, defense, railroads, and even toys. The Department has consistently maintained good ties with industry, and students have had numerous co-op opportunities.

The Computer Science Department currently has thirteen tenured or tenure-track faculty and two lecturers. It has a strong record in terms of teaching and research and will provide a firm foundation in computer science for Software Engineering students. The high quality of the education provided by the Department is demonstrated by qualitative feedback from industry and alums, as well as the quantitative results from ETS Major Field Test results that place the Department in the top 95–99 percentile of all departments across the country that use the test.

The first software engineering course in the Department was introduced at the graduate level when the M.S. degree program in Computer Science was initiated in 1982. A senior level elective was added to the undergraduate curriculum in 1986. In addition, several lower-level required courses have traditionally contained elementary software engineering concepts, in recognition of the fact that many Computer Science graduates enter software engineering

careers.

Several faculty have significant experience in software engineering education, as well as in the discipline of software engineering. One faculty member began research in software measurement working with one of the pioneers of the field in the 1970s. Other relevant areas of faculty research include formal specification, software verification and risk analysis.

7 Institutional Impact

We anticipate incoming classes into the Software Engineering program to approach 50 students within a few years. Of this number, approximately half would have been CS students and half will be students who would not have enrolled at Tech without this program. Considering flows in and out of the program, the total enrollment for all four classes will likely be about 175 students. We anticipate that the software engineering degree program will attract some internal transfers, mostly from electrical and computer engineering and that the transfers out will be a bit more diverse than current computer science students. It is likely that some students may decide that they have become more interested in the area which they were planning to use as their application area.

No new Computer Science courses are required for the program. Thus the impact on the department in terms of teaching will be similar to an increase of about 100 students in our existing program. Additional course sections may be necessary to accommodate these students at the lower levels. In upper level elective courses sufficient space should be available to accommodate these additional students.

The dominant effect on the Department will be from the total number of students that we have in our three degree programs. Most of the courses, faculty, staff and labs will be shared across the three programs. This provides for the greatest economy, but also means that the total number of students enrolled in the Departmental programs must be taken into account when allocating resources. An increase in the total number of students within the Department will require additional faculty and support staff. The additional complexity in advising resulting from having multiple degree programs within the Department is likely to add significantly to the advising load. Depending on total enrollments, a full or part-time advisor for undergraduate students within the Computer Science Department may be the best solution.

Students in this curriculum will be taking a number of credits in mathematics, science and general education courses similar to that of traditional Computer Science students. The effect on these departments will be similar to any increase in enrollments at Tech. Students will be choosing from one of three business courses, thus enrollments in these courses will likely increase. However, the effect will be dispersed among the three courses. Students will also be choosing from a wide range of potential application areas. Unless a particular application area is extremely popular, the impact on courses in the application domains will be minimal.

A primary impact on the institution should be an increase in the number and quality of incoming students. In addition, because there are still few undergraduate programs in

software engineering, the program should attract national interest.

8 Facilities and Equipment

The Department of Computer Science currently has sufficient lab space and computing resources to begin this degree program. These resources include four computer laboratories for undergraduate students, each equipped with twenty workstations. Two labs have Sun workstations running the Solaris operating system, and two labs have PCs running Linux. Students also have access to a variety of compute and file servers. Other departmental facilities include HP and SGI workstations and a network of PCs for distributed computing projects. In addition, the Department has access to the facilities of the Computational Science and Engineering Research Institute which consists of several high performance multiprocessors, including two Beowulf clusters and a Sun Enterprise system. Available software includes standard desktop utilities, a wide range of compilers and development tools, and Rational products for software design and testing.

These existing labs will provide the initial facilities for the Software Engineering program. The Center for Integrated Learning and Information Technology (CILIT) is scheduled to be completed by Fall 2005. This project includes a renovation of and addition to the existing Library, and a new home for the Department of Computer Science, more than doubling our existing space. In addition to offices for computer science faculty and graduate students, the plans for the CILIT include significantly enhanced laboratory facilities for undergraduates in the Computer Science Department. These new labs will be shared by students in the Computer Science program, the proposed Computer Systems Science program and the Software Engineering program. The planned facilities include four computer labs designed for individual work, two additional labs designed for group projects, a computer-equipped undergraduate learning center and study space. In addition, a variety of research computing labs are planned for the CILIT, including visualization and graphics labs, a cluster computing lab, an artificial intelligence and robotics lab, and a computational science and engineering lab; these facilities may be used by Software Engineering students working on senior design projects in related areas. Fund-raising is underway to equip the labs in the new building. Student computing lab fees will be used to maintain and replace the computing equipment and software as needed.

No other new facilities or resources are needed specifically for this program. Current Library resources are sufficient for a software engineering program at the undergraduate level. In particular, the Library has institutional subscriptions to all ACM and IEEE publications, which include the primary periodicals that students are likely to need.

9 Schedule

Because this degree program is based on existing courses, it can be implemented without delay. Thus we plan to initiate the program beginning Fall 2003. All course offerings will be

geared for regular, traditional students. The necessary tasks to fully implement the program can be categorized as follows:

- **ABET accreditation.** Once the program has been in place for some time and produced graduates, we will seek ABET accreditation. In the first years of the program, we will monitor its progress in preparation of eventual accreditation efforts.
- **Initial curricular issues.** We must form a committee to provide oversight of the program curriculum. Course sequences for appropriate application areas must be identified.
- **Outreach to industry and other academic departments.** We must find sponsors of senior design projects, both inside and outside the University. We must establish communication channels with these sponsors for fast and effective feedback in both directions.

10 Curriculum

10.1 Overview

The software engineering curriculum is intended to develop students' knowledge and skills in the following areas:

- **Computer science principles.** Computer scientists have created mathematically sound techniques to build effective, reliable software. Problems can be specified precisely using formal languages. Intermediate designs can be represented and analyzed using established modeling methods. Algorithms can be compared quantitatively in terms of efficiency. These techniques are covered in the Mathematics courses and in Computer Science courses like CS 2311 and CS 4321.
- **Computer science practice.** Of course, practical experience in developing and running software is essential. Students get this experience through programming in Java and C++, starting with the introductory course sequence and CS 2141. The project courses CS 3141 and CS 4790 give more applied hands-on experience. The courses CS 4411 and CS 4421 cover the implementation and use of operating systems and database systems through programming exercises.
- **Software engineering principles and practice.** From the software engineering perspective, software development is a *process*, employing techniques that are well defined and well suited to the application at hand. The software engineering curriculum covers the fundamental components of any development process: capturing *requirements* from the customer; formulating and analyzing a *specification* based on the requirements; choosing and representing intermediate *designs* that meet the specification; *verifying* and *validating* the design to ensure that it meets the specification

and the customer's wishes; *maintaining* previously developed software for future use. These ideas are introduced in the initial Computer Science sequence (CS 1121–1122 or CS 1131–1132) and in CS 2141. CS 4711 and CS 4712 cover them in depth. They are put into practice through the projects in CS 3141, CS 3611 and CS 4790.

- **Application areas.** When producing software aimed at a particular application, developers who have some knowledge in the application area have a distinct advantage; they can communicate more effectively with the customer and design solutions more quickly. The goal of the application electives requirement is to endow students with such domain knowledge. The application electives form a coherent set of courses that provide students some in-depth knowledge in an application area (*e.g.,* business, engineering, science). Each student must take at least one upper-level application elective. The choice of electives must be approved by the student's advisor and the Computer Science Undergraduate Committee.

- **The process of software development.** Experienced software practitioners know that the design and analysis techniques of computer science form only one part of the overall process of software development. The steps toward creating a software product must be carefully planned and scheduled. Teams of developers must be organized, with effective communication channels maintained within and between teams. Customers must be tapped for input and continuously apprised of progress. CS 4711 and the Business & Economics requirement expose students to the management issues surrounding software development. HU 3120 teaches effective communication, perhaps the most important aspect of the process. Finally, CS 3141, CS 3611 and CS 4790 provide experience in team software development.

10.2 Requirements

Computer Science			**53-54**
CS 1000	Orientation	1	
CS 1121, 1122, 2321 (or CS1131 and CS1132)	Computer Science I, II, Data Structures	8-9	
CS 2141	C++ as a Second Language	3	
CS 2311	Discrete Structures	3	
CS 3141	Team Software Project	3	
CS 3421	Computer Architecture	4	
CS 3611	User Interface Design	3	
CS 4000	Senior Seminar	3	
CS 4121	Programming Languages	3	
CS 4321	Algorithms	3	
CS 4411	Intro to Operating Systems	4	
CS 4421	Database Systems	3	
CS 4711	Intro to Software Engineering	3	
CS 4712	Software Quallity Assurance	3	
CS 4790	Senior Design Project	6	
Business & Economics			**3**
BA 3600 or 3780 or 4610	Quality Management or Enterpreneurship or Project Management	3	
Math Courses			**17-18**
MA 1090	Functions, Change and Chance	3	
MA 1160	Calculus with Technology I	4	
MA 2160	Calculus with Technology II	4	
MA 2330	Honors Linear Algebra	3	
MA 2720 or 3710	Statistical Methods or Engineering Statistics	3-4	
Application Electives			**9**
General Education			**40**
UN 1001	Perspectives on Inquiry	3	
UN 1002	World Cultures	4	
UN 2001	Revisions	3	
UN 2002	Institutions	3	
Distribution Courses	(including HU 3120 Technical Communication)	15	
Lab Science		12	
Co-curricular activities		(3 units)	
Electives			**4-6**
TOTAL CREDITS			**128**

10.3 Four-Year Schedule

Year 1

Fall			Spring		
CS 1000	Orientation	1	CS 1122	Intro. to CS II	3
CS 1121	Intro. to CS I	3	MA 1160	Calculus with Tech. I	4
MA 1090	Functions, Change, Chance	3	UN 1002	World Cultures	4
UN 1001	Perspectives on Inquiry	3		Science Elective	4
	Science Elective	4		Co-curricular activity (1 unit)	
	Co-curricular activity (1 unit)				
		14			15

Year 2

CS 2311	Discrete Structures	3	CS 2141	C++ as a Second Lang.	3
CS 2321	Data Structures	3	CS 3421	Computer Architecture	4
MA 2160	Calculus with Tech. II	4	MA 3710	Statistical Methods	3
UN 2001	Revisions	3	UN 2002	Institutions	3
	Science Elective	4		Distribution Elective	3
		17			16

Year 3

CS 3141	Team Software Project	3	CS 3611	GUI Design	3
CS 4411	Intro. to Operating Systems	4	CS 4321	Intro. to Algorithms	3
MA 2330	Honors Linear Algebra	3	CS 4711	Software Engineering I	3
	Application Elective	3		Application Elective	3
	Distribution Elective	3		Distribution Elective	3
				Free Elective	3
		16			18

Year 4

CS 4000	Senior Seminar	3	CS 4421	Database Systems	3
CS 4121	Programming Languages	3	CS 4790	Senior Design Project	3
CS 4712	Software Quality Assurance	3	HU 3120	Tech. and Sci. Comm.	3
CS 4790	Senior Design	3		Application Elective	3
	Business Elective	3		Distribution Elective	3
	Co-curricular activity (1 unit)			Free Elective	2
		15			17

10.4 Course Descriptions

Note: The following information is also available at `http://www.cs.mtu.edu/new/html/classes.html`

CS 1000 Computer Science Orientation Introduction to computer science as a major field of study. Topics include CS options and subfields, career opportunities, the role of computers in society. Students are acquainted, through actual use, with University computing facilities.

CS 1121 Introduction to Computer Science I Starting point of the computer science program. A high-level, object-oriented programming language is introduced as a problem-solving tool. Topics include design, coding, documentation, debugging and testing of programs. Programming assignments are given in both a closed lab setting and as homework. Prerequisite: MA 1032.

CS 1122 Introduction to Computer Science II Continuation of CS1121. Topics include data abstraction, class hierarchies and polymorphism, list, stack and queue data structures, informal complexity-based algorithm and data structure choices, recursion, and an introduction to software development methods. Homework programming assignments are given, including a medium-scale project. Prerequisite: CS 1121.

CS 1131 Computer Science I An alternative starting point of the computer science program for students with some programming experience, combining material from CS 1121 and CS1122 and offering it at an accelerated pace. Homework programming assignments are given. Prerequisite: MA1032; permission of Computer Science Department.

CS 1132 Computer Science II A continuation of CS 1131. Presents material from CS 1121, CS 1122, and CS 2321 at an accelerated pace. Topics include data structures and ADTs (trees, priority queues, dictionaries and graphs) and their implementations, sorting, text processing, and software development concepts. Prerequisite: CS 1131.

CS 2141 C++ as a Second Language This course provides an accelerated introduction to C++ (and C) and an introduction to object oriented design using UML. Topics include C, C++, pointers, virtual functions, use of libraries, object oriented design with UML, structured testing and verification, and object oriented programming in C++. Homework programming assignments are given. Prerequisites: CS 1132 or CS 2321

CS 2311 Discrete Structures Presents fundamental concepts in discrete structures that are used in computer science. Topics include sets, trees, graphics, functions, recursion, proof techniques, logic, combinatorics, formal languages, and machine models. Prerequisites: CS 1122 or CS 1132.

CS 2321 Data Structures Presents fundamental concepts in data structures. Topics include ADTs (trees, priority queues, dictionaries and graphs) and their implementations, algorithm analysis, sorting and text processing. Programming projects are designed to apply these concepts. Prerequisite: CS 1122.

CS 3141 Team Software Project Introduction to the development of large software projects. Presents examples of software design, quality assurance techniques, and test-case design in conjunction with a significant team project, involving design, test, and code documentation, as well as user documentation. Other topics: teamwork, user interfaces, social and professional responsibility. Prerequisites: CS 2311 and (CS 1132 or CS 2321).

CS 3421 Computer Architecture Introduction to the logical structure of computers, including the fundamentals of logic design, information storage and manipulation, control, input/output, and assembly language programming. Topics include a review of current hardware technology, combinational and sequential logic, arithmetic, datapaths, hard-wired control, interrupts, caches, virtual memory, and an introduction to pipelining. Prerequisite: CS 2311.

CS 3611 User Interface Design Principles of the design and implementation of graphical user interfaces. Topics include relation between GUI's and the operating system, hierarchical structure of tools for GUI construction, and human factors in GUI design. Students receive direct experience with the creation of GUI's and are introduced to basic issues of human-machine interaction. Prerequisite: CS 2141.

CS 4000 Senior Seminar Topics include ethical models; legal issues; privacy and security; social responsibility; professional responsibility and service; and the future of computing. Students will complete the ETS MFT assessment exam. Prerequisites: CS3141 and Senior Standing.

CS 4121 Programming Languages A discussion of the concepts underlying programming languages. Topics include programming paradigms: language criteria (including syntax, semantics, run-time behavior, and implementation issues); data, procedure, functional, and control abstraction; functional programming; and logic programming. Prerequisite: CS 2322.

CS 4321 Introduction to Algorithms Fundamental topics in algorithm design, analysis, and implementation. Analysis fundamentals include asymptotic notation, analysis of control structures, solving recurrences, and amortized analysis. Design and implementation topics include sorting, searching, and graph algorithms. Design paradigms include greedy algorithms, divide-and-conquer algorithms, and dynamic programming. Prerequisite: CS 2311 and (CS 1132 or CS 2321).

CS 4411 Introduction to Operating Systems This course presents topics on program representation and execution; operating systems; process and threads; process scheduling; memory management; file systems; network programming; and security and privacy. Prerequisites: CS 2322 and CS 3421.

CS 4421 Database Systems Topics include goals of database management; data definition; data models; data normalization; data retrieval and manipulation; security, integrity, and privacy measures; file, data, and storage organization; object-database systems; and, parallel and distributed databases. Surveys a number of general database systems, and examines in detail at least one database system. Prerequisites: CS 2141 and CS 4411.

CS 4711 Intro to Software Engineering Introduction to software engineering, the study of principled approaches to developing and maintaining software. Topics include software process models, project management, requirements modeling/analysis, design, and testing. Prerequisites: CS 3141.

CS 4712 Software Quality Assurance Building on previous exposure in CS 4711 to the fundamentals of the software process, this course focuses on techniques for ensuring software quality. Topics include formal specification, testing, proof-based verification, reliability models, and metrics for defect prediction. Projects will involve the use of online tools. Prerequisites: CS 4711.

CS 4790 Senior Design Project This course allows students to apply the principles and techniques of software engineering covered in CS4711 and CS4712. Each student will be part of a team responsible for developing a production-quality software product. Prerequisites: CS 4712.

BA 3600 Quality Management Current quality control and management philosophy, concepts and tools: strategic importance, philosophies of leading sages, practices including ISO9000 standards and Baldrige award requirements, process- focused and result-focused tools includes statistical process control. Prerequisites: BA2100 or MA2710 or MA3710.

BA 3780 Entrepreneurship Covers management issues associated with establishing successful new enterprises as small businesses or part of an existing firm. Term project is creating a business plan. Case studies develop opportunity recognition, entrepreneurial teams, reward systems, financing alternatives, family ventures, ethical and legal contractual considerations, and resource needs.

BA 4610 Project Management Application of systems analysis to project definition and selection. Project teams, their structures and interactions. Cross-functional communication in technological project management. Project management planning, scheduling, and control tools. Project monitoring, evaluation, and termination. Multiple project management and

inter-project relations. Case study of new product process development. Requires case study reports.

HU 3120 Technical and Scientific Communication A study of written and oral communication in technical and scientific environments; emphasizes audience, writing processes, genres of scientific and technical discourse, visual communication, collaboration, professional responsibility, clear and correct expression. Students write and revise several documents and give oral report(s). Prerequisites: UN 1002 or UN 1003 or UN 2002.

MA 1090 Functions, Change, and Chance A survey of mathematical ideas and reasoning for computer science majors. Topics may include difference equations and recursion, proof by induction, random number generators and elementary number theory, game trees and strategy, probability, and simulation.

MA 1160 Calculus with Technology I An introduction to single-variable calculus, which includes a computer laboratory. Topics include trigonometric, exponential, and logarithmic functions, differentiation and its uses, and basic integration. Integrates symbolic tools, graphical concepts, data and numerical calculations. Prerequisites: MA 1032 or MA 1033.

MA2160 Calculus with Technology II Continued study of calculus, which includes a computer laboratory. Topics include integration and its uses, function approximation, vectors, and elementary modeling with differential equations. Prerequisites: MA1150 or MA1160 or MA1135.

MA 2330 Honors Elementary Linear Algebra Introduction to linear algebra and how it can be used, including basic mathematical proofs. Topics include systems of equations, vectors, matrices, orthogonality, subspaces, and the eigenvalue problem. Not open to students with credit in MA 2320. Prerequisites: MA 1150 or MA 1160.

MA 2720 Statistical Methods Introduction to the design and analysis of statistical studies. Topics include methods of data collection, descriptive and graphical methods, probability, statistical inference on means, regression and correlation, and single variable ANOVA. Not open to students with credit in MA3710. Prerequisites: MA 1032 or MA 1033.

MA 3710 Engineering Statistics Introduction to the design, conduct, and analysis of statistical studies aimed at solving engineering problems. Topics covered include methods of data collection, descriptive and graphical methods, probability and probability models, statistical inference, control charts, design of experiments. Not open to students with credit in MA2720. Prerequisites: MA2150 or MA2160.

UN 1001 Perspectives on Inquiry Engages students in college level inquiry through which they develop fundamental intellectual habits, understand how to integrate perspectives on knowledge, and begin to learn how to meet the changing needs of a global, technological, diverse, and environmentally sensitive society.

UN 1002 World Cultures Examines diversity and change around the globe from perspectives of social sciences, humanities, and arts; explores human experience from prehistory to present. Classroom lectures accompanied by films, live performances, and guest speakers. One complete year of a single foreign language plus World Cultures (UN1003, 1-credit-activities) substitutes for World Cultures.

UN 2001 Revisions Provides direct instruction in communication and strategies for revision. Writing portfolios provide a starting point for the course. Instruction in the composing process is often accompanied by work in small groups and conferences with the instructor. Prerequisites: UN 1001 and (UN 1002 or UN 1003).

UN 2002 Institutions From families to governments, to markets, to our interactions with the natural environment, institutions organize collective human action. Introduces students to the nature and role of institutions in shaping today's world. Specific topics will vary by section, but all sections address a set of core questions and concepts.

10.5 Course Descriptions for Sample Technical Electives

10.5.1 Biomedical Instrumentation

This application area will prepare students for the complex, safety-critical problems in designing software for biomedical applications.

BL 2010 Anatomy & Physiology I Comprehensive introductory course in vertebrate anatomy and physiology with emphasis on the human body. Interrelates structure with function in regard to maintaining homeostasis and normal functioning of the body. Covers the integument, skeletal system, nervous system, muscles, and the endocrine system. Prerequisites: CH 1110 or CH 1100.

BL 2011 Anatomy & Physiology I Lab The laboratory to accompany BL2010. Examines embryology, muscle and skeletal anatomy, and neuroanatomy. Explores the physiology of the nervous system, including vision and reflexes and muscle physiology. A student-designed lab project is used to teach experimental design. Prerequisites: BL 2010(C).

EE 3010 Circuits and Instrumentation Designed for nonmajors. Covers the principles of electrical and electronic measurements, including dc, ac, semiconductor devices, amplifiers, and filtering.

BE 3600 Biomedical Instrumentation Introduction to theory of measurement and analysis from biological systems. Covers the use of transducers, data recording and analysis systems and signal processing techniques. Laboratory includes measurements of physiological quantities from living systems. Prerequisites: EE 3010.

10.5.2 Computational Chemistry

With the domain knowledge they acquire from this application area, students will be able to develop software to model chemical processes.

CH 2400 Principles of Organic Chemistry Discusses properties and reactions of various functional groups using reaction mechanisms as a unifying theme. Emphasizes practical applications using industrial, environmental, current events, and biological/medicinal examples. Prerequisites: CH 1120.

CH 3500 Physical Chemistry for Environmental and Life Sciences Equilibrium thermodynamics, chemical kinetics, transport properties, gas laws, and phase equilibria with an emphasis on solution behavior and applications to molecules important in the environmental and life sciences. Prerequisites: (CH 1100 or CH 1110) and (CH 1120 or CH 1140) and (MA 2150 or MA 2160)

CH 5560 Computational Chemistry Focuses on the theory and method of modern computational techniques applied to the study of molecular properties and reactivity through lecture and computer projects. Covers classical mechanical as well as quantum mechanical approaches. Prerequisites: CH 3500 or 3520.

10.5.3 Operations Management

Software plays an important role in the planning and scheduling aspects of manufacturing and distribution. Students choosing this application area will have a good understanding of the challenges and solutions in this domain.

BA 3610 Operations Management Fundamental principles of operations and service management; includes strategic importance and relevant interrelated concepts and tools in product/process design, work systems, forecasting, inventory and materials management, just-in-time, scheduling, capacity management, and maintenance management. Prerequisites: BA 2100 or MA 2710 or MA 3710.

BA 3800 Principles of Marketing Emphasizes decisions made in developing both strategic and tactical marketing plans. Uses computer simulations, experiential learning assignments, and marketing plan development to demonstrate principles of market segmentation, product development, pricing, distribution planning, and promotion.

BA 4620 - Supply Chain Management Designing and managing channels of distribution, purchase and movement of goods, and transportation systems. Emphasizes design of appropriate marketing channels, advanced topics in inventory control, facility location, routing of physical flows among facilities, and design and evaluation of transportation systems. Prerequisites: BA 3600 and BA 3800.

References

[1] Bureau of Labor Statistics (2002). *Occupational Outlook Handbook.* U.S. Department of Labor. Report on "computer software engineers" available at `http://www.bls.gov/oco/ocos267.htm`

[2] Gannon University. Software Engineering Program. `http://www.ecs.gannon.edu/CIS/se.html`

[3] D.D. Grant (2000). Undergraduate software engineering degrees in Australia. In *Proc. IEEE Conference on Software Engineering Education and Training.*

[4] J.V. Harrison (1997). Enhancing software development project courses via industry participation. In *Proc. IEEE Conference on Software Engineering Education and Training.*

[5] A.J. Kornecki, I. Hirmanpour and M. Towhidnajad (1997). Strengthening software engineering education through academic industry collaboration. In *Proc. IEEE Conference on Software Engineering Education and Training.*

[6] J. Margolis, A. Fisher and F. Miller (2002). *Unlocking the Clubhouse: Women in Computing.* MIT Press.

[7] J. Margolis, A. Fisher and F. Miller (1999). Caring About Connections: Gender in Computing. *IEEE Technology and Society Magazine* 18(4): 13–20.

[8] H. Mayr (1997). Teaching software engineering by means of a "virtual enterprise". In *Proc. IEEE Conference on Software Engineering Education and Training.*

[9] M. McCracken, I. Hsi, H. Richter, R. Waters and L. Burkhart (1998). A proposed curriculum for an undergraduate software engineering degree. In *Proc. IEEE Conference on Software Engineering Education and Training.*

[10] Milwaukee School of Engineering. Software Engineering home page. `http://www.msoe.edu/eecs/se/`

[11] Mississippi State University. Bachelor of Science in Software Engineering Degree Requirements. `http://www.cs.msstate.edu/UNDERGRADUATE/sedegreeRequirements.html`

[12] J.F. Naveda and M.J. Lutz (1997). Crafting a baccalaureate program in software engineering. In *Proc. IEEE Conference on Software Engineering Education and Training.*

[13] D.L. Parnas (1998). Software engineering programmes are not computer science programmes. *Annals of Software Engineering* 6 (1/4), 19–37.

[14] A. Parrish, R. Borie, D. Cordes, B. Dixon, D. Hale, J. Hale, J. Jackson and S. Sharpe (1998). Computer engineering, computer science and management information systems: Partners in a unified software engineering curriculum. In *Proc. IEEE Conference on Software Engineering Education and Training.*

[15] Rochester Institute of Technology. Software Engineering home page. `http://www.cs.rit.edu/~softeng/`

[16] M.J. Sebern and M.J. Lutz (2000). Developing undergraduate software engineering programs. In *Proc. IEEE Conference on Software Engineering Education and Training.*

[17] University of Michigan, Dearborn. Bachelor of Science in Software Engineering. `http://www.engin.umd.umich.edu/CIS/bsse/degree.html`