Board of Regents, State of Iowa

REQUEST TO IMPLEMENT A NEW BACCALAUREATE, MASTERS, DOCTORAL OR FIRST PROFESSIONAL DEGREE PROGRAM

THE PURPOSE OF ACADEMIC PROGRAM PLANNING: Planning a new academic degree program provides an opportunity for a Regent University to demonstrate need and demand as well as the university’s ability to offer a quality program that is not unnecessarily duplicative of other similar programs offered by colleges and universities in Iowa.

Institution: Iowa State University

Departments involved: Electrical and Computer Engineering Department
Computer Science Department

CIP Discipline Specialty Title: Software Engineering

CIP Discipline Specialty Number (six digits): CIP 14.0903

Level: B

Title of Proposed Program: Curriculum in Software Engineering leading to the degree Bachelor of Science

Degree Abbreviation (e.g., Minor, B.S., B.A., M.A.): B.S

Approximate date to establish degree: Month August Year 2007

Contact person(s): (name, telephone, and e-mail) Arun Somani (294-0442, arun@iastate.edu), Carl Chang (294-6516, chang@iastate.edu), or Suraj C. Kothari (294-7212, kothari@iastate.edu)

Please provide the following information (use additional pages as needed).

1. Describe the proposed new degree program, including the following:
   a. A brief description of the program and a statement of objectives including the student learning outcomes and how the learning outcomes will be assessed;

   Software Engineering is aimed at creating high-quality software in a systematic, controlled, and efficient manner. The specific objective of the program is to educate students on principles, processes, techniques, and tools for producing, analyzing, specifying, designing and evolving software. A broader objective is to cultivate among students intellectual curiosity, problem solving skills, good learning habits, effective communication skills, leadership, and teamwork.

   The expected learning outcomes:
   1. show mastery of the software engineering knowledge and skills, and professional issues necessary to begin practice as a software engineer;
   2. work as an individual and as part of a team to develop and deliver quality software;
3. reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations;

4. design appropriate solutions in one or more application domains using engineering approaches that integrate ethical, social, legal, and economic concerns;

5. demonstrate an understanding of and apply current theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification and documentation;

6. demonstrate an understanding and appreciation for the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment; and

7. learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.

Outcome assessments will include regular and on-going course and curriculum assessment, based on student course evaluations, published benchmark standards, examination results, feedback from industrial employers, and faculty review of the curriculum. Both the feedback from these sources and the revisions implemented as the result of the feedback will be documented.

b. The relationship of the proposed new program to the institutional mission and how the program fits into the institution’s, college’s, and department/program’s strategic plan;

The proposed program is important for Iowa State University to remain at the forefront of engineering disciplines by providing education in a key emerging area of engineering. The two key professional organizations IEEE/ACM emphasize the need for degree programs in software engineering. In May 2004, ACM and IEEE issued a joint report on model curriculum for developing undergraduate degree programs in software engineering. ABET, the key organization that accredits engineering programs, has recognized software engineering as a separate engineering discipline and established accreditation guidelines for it. The proposed program complements the existing Computer Engineering and Computer Science programs. Establishing this program is a high priority decision for the Electrical and Computer Engineering and Computer Science departments.

c. The relationship of the proposed new program to other existing programs at the institution; describe how the proposed program will enhance other programs at the university.

The proposed program complements the existing Computer Engineering and Computer Science programs. Software has become a critical part of infrastructure for medical, energy, transportation, and financial applications. Industry is experiencing an acute shortage of qualified software engineers and the need for such engineers continues to grow. The proposed program allows focusing on education on engineering aspects of developing and evolving complex software.

d. The relationship of the proposed new program to existing programs at other colleges and universities in Iowa, including how the proposed program is different or has a different emphasis than the existing programs;
This will be the first B.S. degree program in software engineering in Iowa. Computer science and computer engineering are related programs offered by the Iowa State University and the University of Iowa. The University of Northern Iowa offers computer science.

The specifics of the curriculum are different. The professional organizations ACM, IEEE, and the accreditation board ABET consider Computer Science, Computer Engineering, and Software Engineering as different degree programs and specify different model curriculums for them. Unlike the existing program in computer engineering and computer science, the software engineering program will specifically focus on education on engineering aspects of developing and evolving complex software.

e. Special features or conditions that make the institution a desirable, unique, or appropriate place to initiate such a degree program.

Iowa State University has been strong in engineering education and research. The Electrical and Computer Engineering (ECE) and the Computer Science (CS) are strong and growing departments at ISU.

f. Does the proposing institution have personnel, facilities, and equipment adequate to establish and maintain a high quality program?

Yes. The ECE and CS departments have significant resources in terms of faculty, facilities, and equipment. Both the departments have aggressively hired faculty in areas related to software engineering. The ECE department has received seed funding from industry to establish a strong software engineering program. A software engineering lab has been set up with the help of industry funding.

g. How does student demand for the proposed program justify its development?

There is a strong student demand fuelled by a strong job market.

2. Describe the state and/or national workforce need and/or demand for graduates of the proposed program currently and in the near future (provide documentation about the sources of data used to estimate need and demand.)


3. List all other public and private institutions of higher education in Iowa currently operating programs similar to the proposed new degree program. (For comparison purposes, use a broad definitional framework, e.g., such identification should not be limited to programs with the same title, the same degree designation, having the same curriculum emphasis, or purporting to meet exactly the same needs as the proposed program.)

This will be the first B.S. degree program in software engineering in Iowa. Computer science and computer engineering are related programs offered by the Iowa State University and the University of Iowa. The University of Northern Iowa offers computer science. The letters from the other two institutions are attached
If the same or similar program exists at another public or private institution of higher education in Iowa, respond to the following questions:

a. Could the other institution reasonably accommodate the need for the new program through expansion? Through collaboration?

b. With what representatives of these programs has there been consultation in developing the program proposal? Provide a summary of the response of each institution consulted.

c. Has the possibility of an inter-institutional program or other cooperative effort been explored? What are the results of this study? (Consider not only the possibility of a formally established inter-institutional program, but also how special resources at other institutions might be used on a cooperative basis in implementing the proposed program solely at the requesting institution.)

4. Estimate the number of majors and non-majors students that are projected to be enrolled in the program during the first seven years of the program.

a. Undergraduate

The focus will be on majors because of limited resources including the faculty, teaching assistants, and lab facilities with limited number of seats.

The estimates are expected to vary within 10% to account for dropouts. We do not expect significant number of dropouts. However, if that happens we can recruit more students in subsequent years. Given the job opportunities for software engineers and their limited supply, we will be able to sustain this level of enrollment.

The estimates are based on the resources we expect to have to support the program; we can enroll 48 new students every year. Given the demand in the industry, more students are likely to be interested in the program, but we expect to control the number because of limited resources (personnel and laboratory facilities)

<table>
<thead>
<tr>
<th>Undergraduate</th>
<th>Yr 1</th>
<th>Yr 2</th>
<th>Yr 3</th>
<th>Yr 4</th>
<th>Yr 5</th>
<th>Yr 6</th>
<th>Yr 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majors</td>
<td>24</td>
<td>48</td>
<td>120</td>
<td>144</td>
<td>192</td>
<td>192</td>
<td>192</td>
</tr>
<tr>
<td>Non-Majors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. Graduate

Not applicable.

<table>
<thead>
<tr>
<th>Graduate</th>
<th>Yr 1</th>
<th>Yr 2</th>
<th>Yr 3</th>
<th>Yr 4</th>
<th>Yr 5</th>
<th>Yr 6</th>
<th>Yr 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-Majors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

c. What are the anticipated sources of these students?

We expect that some of the current computer science and computer engineering students may move to the software engineering program. However, it is expected that more than half of the students will be new student who otherwise would not have come to ISU. Thus the two departments will attract new students specifically interested in software engineering.
5. If there are plans to offer the program away from the campus, briefly describe these plans, including potential sites and possible methods of delivery instruction.
   No such plans.

6. Has the proposed program been reviewed and approved by the appropriate campus committees and authorities? List them:
   Electrical and Computer Engineering Curriculum Committee
   Computer Science Curriculum Committee
   Engineering College Curriculum Committee
   LAS College Curriculum Committee
   Faculty Senate Curriculum Committee
   Faculty Senate Council on Academic Affairs
   Faculty Senate

7. List date the program proposal was submitted to the Iowa Coordinating Council for Post High School Education (ICCPHSE) and the results of listserv review. (THIS WILL BE FILLED IN BY THE PROVOST OFFICE.)

8. Will the proposed program apply for accreditation? When?
   Yes. ABET (http://www.abet.org/) accredits undergraduate programs in software engineering. We expect to follow a six-year cycle. The year for the first visit is to be determined.

9. Will articulation agreements be developed for the proposed program? With whom?
   Yes, if possible. So far, we have approached DMACC and they are interested. We are working through the College of Engineering Dean’s office to explore the possibilities.
10. Describe the faculty, facilities, and equipment that will be required for the proposed program. From where will the financial resources for the proposed program come (list all that apply, e.g., department reallocation, college reallocation, grants, new to the university)?

We expect to run the program with 8 faculty members, 10 teaching assistants, and two laboratories.

<table>
<thead>
<tr>
<th>SOURCES</th>
<th>TOTAL AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department Reallocation (ECE and CS will contribute equal resources. Together they will contribute 6 FTE, 6 Teaching Assistants</td>
<td>$690,000</td>
</tr>
<tr>
<td>New Resources from College and University: 2 new FTE and 4 teaching assistants, advising/computer support staff.</td>
<td>$310,000</td>
</tr>
<tr>
<td>Departments and External Funding: New Software Engineering Labs</td>
<td>$300,000</td>
</tr>
</tbody>
</table>

The 6 FTE include the two new faculty hired into related areas in 2005.

Industry has contributed seed funding, hardware, and software. The ECE department established a new software engineering laboratory in 2005 with funding and software from industry.

The new resources (as described in the next table) will come from the two departments and the colleges. See the attached letters.

11. Estimate the total costs/total new costs (incremental increases each year in expenditures) that will be necessary for the next seven years as a result of the new program:

<table>
<thead>
<tr>
<th>Year</th>
<th>TOTAL COSTS</th>
<th>TOTAL NEW COSTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year 1</td>
<td>$990,000</td>
<td>None</td>
</tr>
<tr>
<td>Year 2</td>
<td>$1,000,000</td>
<td>$310,000</td>
</tr>
<tr>
<td>Year 3</td>
<td>$1,000,000</td>
<td>None</td>
</tr>
<tr>
<td>Year 4</td>
<td>$1,000,000</td>
<td>None</td>
</tr>
<tr>
<td>Year 5</td>
<td>$1,100,000</td>
<td>$100,000</td>
</tr>
<tr>
<td>Year 6</td>
<td>$1,000,000</td>
<td>None</td>
</tr>
<tr>
<td>Year 7</td>
<td>$1,000,000</td>
<td>None</td>
</tr>
</tbody>
</table>

Budget Explanation:

Year 1: 6 FTE with an average of $100,000 per FTE; salaries for 6 teaching assistants an average of $15,000 per teaching assistant, New Laboratories $300,000.
Year 2: Salary expenses as in year 1 plus two new FTE and 4 new teaching assistants, and advisor or computer support staff salary expenses of $50,000.
Year 5: $100,000 for renovating the laboratories.
Supplemental materials
(to be used at Iowa State University in the review of the proposal):

13. Program requirements, including:
   a. prerequisites for prospective students;
   b. language requirements;
   c. courses and seminars presently available for credit toward the program;
   d. proposed new courses or modifications of existing courses;
   e. thesis and non-thesis options in master's programs;
   f. implications for related areas within the university;
   g. admissions standards for graduate programs

We have attached a detailed proposal that includes the above information.

14. Attach to the program proposal memos from the department chair(s), the college dean(s), and other appropriate persons, agreeing to the allocation of new resources and/or the reallocation of resources as described in the Regents questions

Letters are attached.

15. Attach to the program proposal, letters of support, recommendations, and statements when appropriate:
   a. from programs at the other Regents universities
   b. from programs and departments at ISU which are associated with the proposed program or have an interest in the proposed program

Letters from University of Iowa and University of Northern Iowa are attached.

Notes:

1. The CIP Specialty name and number is found at:
Departments Originating the Proposal
1. Department of Electrical and Computer Engineering, Iowa State University
2. Department of Computer Science, Iowa State University

Name of Degree
B.S. in Software Engineering

Departments Administering the Program
The degree program will be jointly administered by the Department of Electrical and Computer Engineering (ECE) and the Department of Computer Science (CS). The students will be able to enroll through either department. By working together we will avoid any duplication of resources required to teach new courses for the SE program. The ECE and the CS departments will share equally the responsibility of teaching new courses required for the SE Program. A joint Software Engineering Program Administration Committee with equal representation from both the departments will be set up for administering the program.

Contact Persons
Suraj C. Kothari and Arun Somani, Department of Electrical and Computer Engineering, Robyn R. Lutz and Carl K. Chang, Department of Computer Science

The Software Engineering Discipline
Software engineering is “the application of a systematic, disciplined, quantifiable approach to the development, operations, and maintenance of software” [1].

The key knowledge areas needed to learn software engineering within a university undergraduate program have been recently specified in “Computing Curricula—Software Engineering” (CCSE) [1], produced by the Joint Task Force on Computing Curricula under the direction and guidance of the two relevant preeminent professional organizations, the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE), and the Association for Computing Machinery (ACM). That CCSE report on software engineering education forms the basis for this proposal.

As stated by [1], “Software Engineering is about creating high-quality software in a systematic, controlled, and efficient manner. Consequently, there are important emphases on analysis and evaluation, specification, design and evolution of software. In addition, there are issues related to management and quality, to novelty and creativity, to standards, to individual skills, and to teamwork and professional practice that play a vital role in software engineering.”

Need for Software Engineers
Industry is experiencing an acute shortage of qualified software engineers and the need for such engineers continues to grow. Despite outsourcing, the US Bureau of Labor Statistics is currently projecting a 10-year increase in US programmer positions of a healthy 45-46% (down from the >90% projection of the bubble period) [3]. The yearly increase in programmer productivity, estimated at several percent per year, is significantly lower than the rates of improvement for major hardware components like chips, secondary storage, networks and displays. This suggests software engineering will likely be a continuing bottleneck in computing, leading to a long term need for software engineering education. Thus, software engineering instruction is and will continue to be important.

In the past decade, the demand for software engineers grew to the extent that the industry was forced to hire even people with little or no formal training in software development. The demand for new software engineers already exceeds the supply, a gap that is only expected to widen as we attempt to both produce more software and software for more challenging systems.
According to a 2001 survey funded by ACM and IEEE, there are 60 universities in 11 countries that offer SE programs, with 6,799 students enrolled in those programs. The major engineering accreditation organization ABET considers software engineering to be a key engineering discipline and has established accreditation guidelines for it. We believe it is necessary for Iowa State University to introduce such a program to remain at the forefront of engineering & computing education.

**Objectives**

The objective of the Software Engineering program at ISU is that its graduates should demonstrate expertise, engagement, learning, leadership, and teamwork within five years after graduation.

- **Expertise:** Graduates should establish peer-recognized expertise together with the ability to articulate that expertise and use it for problem solving in the planning, design, development, validation, and evolution of software using contemporary practices.
- **Engagement:** Graduates should be engaged in the professional practice, locally and globally, contributing through the ethical, competent, and creative practice of Software engineering in industry, academia, or the public sector, or graduates may use the program as a foundation for interdisciplinary careers in business, law, medicine, or public service.
- **Learning:** Graduates should demonstrate sustained learning through graduate work or professional improvement opportunities and through self study, and they should demonstrate the ability to adapt to rapid technological changes.
- **Leadership:** Graduates should exhibit leadership and initiative to advance professional and organizational goals, facilitate the achievements of others, and obtain results.
- **Teamwork:** Graduates should demonstrate effective teaming and commitment to working with others of diverse cultural and interdisciplinary backgrounds by applying software engineering abilities, communication skills, and knowledge of contemporary and global issues (Details in Appendix A).

**Outcomes**

The Joint Task Force on Computing Curricula Software Engineering has identified a number of desirable outcomes for an undergraduate curriculum in software engineering [1]. According to that list, graduates of an undergraduate software engineering program must be able to:

8. show mastery of the software engineering knowledge and skills, and professional issues necessary to begin practice as a software engineer;
9. work as an individual and as part of a team to develop and deliver quality software;
10. reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations;
11. design appropriate solutions in one or more application domains using engineering approaches that integrate ethical, social, legal, and economic concerns;
12. demonstrate an understanding of and apply current theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification and documentation;
13. demonstrate an understanding and appreciation for the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment; and
14. learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.
The Joint Task Force on Computing Curricula Software Engineering has identified these outcomes as specific to software engineering; these compliment the general ABET outcome requirements applicable to all engineering programs [6].

We adopt these outcomes as central to our proposed B.S. in Software Engineering. Outcome assessments will include regular and on-going course and curriculum assessment, based on student course evaluations, published benchmark standards, examination results, feedback from industrial employers, and faculty review of the curriculum. Both the feedback from these sources and the revisions implemented as the result of the feedback will be documented.

**Software Engineering Curriculum**

The curriculum described below categorizes the courses and follows the guidelines as proposed by the CCSE Report [1] (Course number is accompanied with credit hrs).

- **ComS/CprE Introductory courses**: ComS 227 (4), ComS 228 (3), CprE 210 (4), CprE 211 (4)/ComS 229(3)
  - *Total Credit Hrs: 15/14 (choice of CprE 211 includes embedded systems labs, adds 1 credit hour)*

- **ComS/CprE Intermediate courses**: ComS 311 (3), CprE 308 (4)/ComS 352 (3), ComS 363 (3)
  - *Total Credit Hrs: 10/9 (choice of CprE 308 includes Linux labs, adds 1 credit hour)*

  - *Total Credit Hrs: 20*

- **Software Engineering Electives**: Two software engineering electives from the following list:
  - SE 409: Software Requirements Engineering
  - SE 412: Formal Aspects of Specification and Verification
  - SE 416: Software Evolution and Maintenance
  - SE 417: Software Testing
  - *Total Credit Hrs: 6*

- **Mathematics courses**: CprE 310 (3)/ComS 330(3), Stat 330 (3), Math 165 (4), Math 166 (4), Math 267 (4) and one of the following four courses: Math 304 (3), Math 307 (3), Math 314 (3), Math 317 (3)
  - *Total Credit Hrs: 21*

- **Communication and Economics courses**: Econ 101/102 or IE 305 (3), SpCm 212 (3), Engl 104 (3), Engl 105 (3), Engl 314 (3)
  - *Total Credit Hrs: 15*

- **Library and Natural Science courses**: Lib 160 (0.5), Phy 221 (5), Chem 167 (4) (An appropriate biology course will be a preferable alternative to Chem 167, if and when the basic engineering requirements are changed by the College of Engineering to allow such an alternative)
  - *Total Credit Hrs: 9.5*
- **Technical Electives**: One SE/ComS/CprE 400 level course to be selected in consultation with SE advisors.
  - *Total Credit Hrs: 3 credit hours*

- **Supplementary/General Education Courses**: These courses to be selected in consultation with SE advisors. They must include courses to satisfy university and college general education requirements. These courses include courses, or categories of courses, that may need to be met, such an international perspectives, U.S. diversity, language, arts and humanities, and social sciences. The courses should also include electives.
  - *Total Credit Hrs: 25-27 credits.*

- **Professional Orientation**: Engr 101, Orientation (0), CprE 166, Professional Orientation (0) or equivalent ComS professional orientation courses.
  - *Total Credit Hrs: 0*

- **Total credits in Software Engineering Program**: 124.5

### Table 2: Comparison with a Basic Model in the CCSE-2004 Report

<table>
<thead>
<tr>
<th>Course type</th>
<th>IEEE-ACM Model</th>
<th>Proposed Curriculum</th>
<th>ISU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComS/CprE Introductory Courses</td>
<td>9</td>
<td>15-14</td>
<td></td>
</tr>
<tr>
<td>ComS/CprE Intermediate Courses</td>
<td>9</td>
<td>10-9</td>
<td></td>
</tr>
<tr>
<td>Software Engineering Courses (Core + Electives)</td>
<td>21</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Software Engineering 2-semester Capstone Course</td>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Technical Electives</td>
<td>12</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Math Courses</td>
<td>15</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>Communication and Economics Requirement</td>
<td>9</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>81</strong></td>
<td><strong>89/87</strong></td>
<td></td>
</tr>
</tbody>
</table>

The CCSE Report [1] recommends a general structure of courses for an undergraduate degree program in software engineering. Table 2 shows how the proposed ISU curriculum compares with the CCSE curriculum. The proposed curriculum has stronger mathematics and communication requirements. We also provide more flexibility by reducing the number of required technical electives from 12 to 6 but including 27 additional credits of electives. A student may choose additional technical courses for depth, or other courses for breadth, or courses to acquire a minor in another area.

### Comparison with Accredited Software Engineering BS programs

Six universities in the U.S. – Clarkson University, Milwaukee School of Engineering, Florida Institute of Technology, Mississippi State University, Rochester Institute of Technology and University of Texas at Arlington – offer software engineering BS programs that have been accredited by ABET.

Clarkson University: ([http://www.clarkson.edu/ece/handbook/software.html](http://www.clarkson.edu/ece/handbook/software.html))  
Milwaukee School of Engineering ([http://www.msoe.edu/eecs/se/proginfo](http://www.msoe.edu/eecs/se/proginfo))  
Rochester Institute of Technology ([http://www.rit.edu/~932www/ugrad_bulletin/colleges/ccis/softeng.html](http://www.rit.edu/~932www/ugrad_bulletin/colleges/ccis/softeng.html))

The following table summarizes the comparison between these programs and the proposed program at ISU:
<table>
<thead>
<tr>
<th>School</th>
<th>Core Courses</th>
<th>Senior Projects</th>
<th>Math</th>
<th>Communication</th>
<th>Economics</th>
<th>Total Credit Hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iowa State University (pending)</td>
<td>6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>124.5</td>
</tr>
<tr>
<td>Clarkson University (2003)</td>
<td>--</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>121(CE)/122(CS)</td>
</tr>
<tr>
<td>Milwaukee School of Engineering (2003)</td>
<td>--</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>194*</td>
</tr>
<tr>
<td>Rochester Institute of Technology (2003)</td>
<td>6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>194*</td>
</tr>
</tbody>
</table>

* Quarter system

The complete curriculum of the proposed B.S. degree in Software Engineering is provided in Appendix A. The descriptions of the new software engineering courses are given in Appendix B. The learning objectives and outcomes of the two-semester capstone project sequence will be similar to the current two-semester senior design project sequence of the Computer Engineering program at ISU, but with an emphasis on software techniques, process and practice.

**Implications for Related Degrees**

The proposed SE degree program will attract students from ComS and CprE programs who have strong interests in engineering large software systems. As the SE program will offer what is currently a hard-to-find educational opportunity, it will also attract some highly motivated undergraduate students who would not have opted for ISU otherwise.

**Resource Needs**

The Software Engineering Program is designed to graduate approximately 48 students per year once it is fully implemented. Eight new sections plus the 2-semester capstone project will be taught each year. The eight sections include four core courses and four electives. Resource requirements are as follows:

1. **Faculty:** Two additional faculty positions (six new courses will be introduced: see Deployment plan below. Average course load for each faculty is 3 per year. Existing faculty members already carry full course loads in Computer Science and Computer Engineering department.)
2. **TAs:** Fourteen additional TA positions (two for each of SE 185 (software engineering problem solving course), 319, 329, 339 and one each for SE 441, 442, 409, 412, 416, 417)
3. **Staff:** One additional computer support staff member
4. **Laboratory:** Two teaching laboratories with best-practice software tools and development environments.

In order to successfully implement, sustain, and enhance the program, well-qualified faculty and TA support and well-equipped laboratories are a must. Startup costs are: travel funds for visiting successful software engineering programs for lessons learned; travel funds to solicit industrial collaborations; and faculty release time of one course per year over the next three years to establish the program.
Existing Faculty

Faculty members in the Department of Computer Science and the Department of Electrical and Computer Engineering associated with Software Engineering Program are as follows:

*Department of Computer Science*: Samik Basu, Carl K. Chang, Gary T. Leavens, Markus Lumpe, Robyn R. Lutz, Hridesh Rajan and Andrew S. Miner. These faculty members carry full loads in Computer Science.

*Department of Electrical and Computer Engineering*: Suraj C. Kothari, Tien Nguyen, G. Manimaran, Y. Guam, T. Daniels, and Ratnesh Kumar. These faculty members carry full loads in ECE Department.

Deployment (Tentative Plan)

One core course and one elective were introduced on an experimental basis in spring, 2004. Six new courses will be introduced over the two years.

1. Fall 2004: two electives (SE 409, SE 416)
2. Spring 2005: one elective (SE 417)
3. Fall 2005: two core courses (SE 319 and SE 329) and the first semester capstone (SE 441).
4. Spring 2006: one core (SE 339), and the second semester capstone (SE 442).
5. Spring 2006: SE 185

NOTE: We plan to introduce the SE courses initially as ComS/CprE experimental courses. Since the first version of this proposal was circulated in 2004, we have introduced five of these courses.

The contents of these courses will be refined after the first offering, if necessary. The course offerings will become stable starting with the fall 2006. There will be two core and two elective software engineering courses each semester. The two-semester capstone project will be offered continuously on both a fall-spring and spring-fall basis. Assuming the program proceeds according to the above tentative plan the first B.S. degree in software engineering may be granted as early as the spring of 2006.

Collaboration with Industry

The industry is highly supportive of our efforts in educational and research activities in software. We are engaged in various collaborations with them and the level of our interaction is growing.

- **Partnerships in education:** formalize contacts to get feedback on curriculum, get case studies and other material to enrich courses, arrange guest lectures and mentorship, and team teach during the first offering of selected courses.
- **Partnerships in research:** establish industry funded research projects, and pursue joint projects funded by federal agencies like DARPA, NSF, and NASA.
- **Other partnerships:** arrange cooperative programs and internships, scholarships, distance education, colloquiums, graduate student fellowships, summer fellowships for faculty, gifts from industry foundations, and chaired professorships in software engineering.

Lockheed Martin, Rockwell Collins, and Guidant have provided seed funding to the Department of Electrical and Computer Engineering to promote software engineering education and research. IBM has provided state-of-the-art Rational software tools for setting up an educational software engineering laboratory.
References


## Appendix A: Proposed SE Curriculum

<table>
<thead>
<tr>
<th>Year</th>
<th>Cr</th>
<th>First semester (1)</th>
<th>Cr</th>
<th>Second semester (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>Math 165: Differentiation</td>
<td>4</td>
<td>Math 166: Integration</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Engl 104: Composition I</td>
<td>3</td>
<td>Engl 105: Composition II</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>SE 185 (SE Problem Solving)</td>
<td>5</td>
<td>Phys 221: Mechanics, elect. flds, DC ccts.</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>Library 160: Library instruction</td>
<td>3</td>
<td>ComS 228: Programming-2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>ComS 227: Programming-1</td>
<td>R</td>
<td>ComS 203/CprE 166: Professional Orientation</td>
</tr>
<tr>
<td></td>
<td>15.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Cr</th>
<th>First semester (3)</th>
<th>Cr</th>
<th>Second semester (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>Math 267: ODEs, matrices, lin. eqs., LaPlxf</td>
<td>3</td>
<td>Math Elective</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Supplementary/Gen. Ed. Electives</td>
<td>3</td>
<td>ComS 330/CprE 310: Discrete Comp. Structures</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>CprE 210: Intro to digital design</td>
<td>3</td>
<td>SE 309 Software Development Practices</td>
</tr>
<tr>
<td></td>
<td>4/3</td>
<td>CprE 211(4)/ComS 229 (3)</td>
<td>3</td>
<td>Economics Course</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Chem 167</td>
</tr>
<tr>
<td></td>
<td>15/14</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Cr</th>
<th>First semester (5)</th>
<th>Cr</th>
<th>Second semester (6)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>Supplementary/Gen. Ed. Electives</td>
<td>3</td>
<td>ComS 363: DBMS</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>ComS 311: Design and Analysis of Algorithms</td>
<td>3</td>
<td>SE 339 Software Architecture and Design</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>SE 319 Software Construction and User Interface</td>
<td>3</td>
<td>Engl 314: Technical communication</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>SE 329: Software Project Management</td>
<td>6</td>
<td>Supplementary/Gen Ed. Electives</td>
</tr>
<tr>
<td></td>
<td>4/3</td>
<td>CprE 308(4)/ComS 352(3)</td>
<td>3</td>
<td>SP CM 212: Public Speaking</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16/15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Cr</th>
<th>First semester (7)</th>
<th>Cr</th>
<th>Second semester (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>Stat 330: Prob. &amp; Stats. for Computer Science</td>
<td>2</td>
<td>SE 442 Capstone project II</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>SE 441: Capstone project I</td>
<td>3</td>
<td>Software Engineering Elective</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Software Engineering Elective</td>
<td>10/12</td>
<td>Supplementary/Gen Ed. Electives</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Supplementary/Gen Ed. Electives</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Technical Elective</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15/17</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Cr</th>
<th>First semester (7)</th>
<th>Cr</th>
<th>Second semester (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>SE 442 Capstone project II</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>Software Engineering Elective</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10/12</td>
<td>Supplementary/Gen Ed. Electives</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
<td>15/17</td>
<td></td>
</tr>
</tbody>
</table>

|      | Total Credits | 125.5 |

- The students who take the 4-credit lab courses CprE 211 and/or CprE 308 instead of the corresponding 3-credit alternatives can apply the additional credits towards Supplementary/General Educational Electives. Thus, the total number of required credit remains the same for all students.

http://www.iastate.edu/%7eregistrar/courses/diversity-list.htm,
http://www.iastate.edu/~registrar/courses/internat.html.
Appendix B: New Software Engineering Courses

Software Core Courses
  SE 185: Problem Solving
  SE 309: Software Development Practices
  SE 319: Software Construction and User Interfaces
  SE 329: Software Project Management
  SE 339: Software Architecture and Design

Software Engineering Electives – 400-level Courses
  SE 409: Software Requirements Engineering
  SE 412: Formal Aspects of Specification and Verification
  SE 416: Software Evolution and Maintenance
  SE 417: Software Testing

Capstone Project in Software Engineering
  SE 441
  SE 442
Descriptions of Software Engineering Courses

SE 185: Problem Solving in Software Engineering

Catalog Description

Learning objectives
This is the first course in software engineering. The students are expected to learn methodical problem solving in the context of software engineering. Specifically, students are expected to learn:

- Formulating the problem – requirement analysis in software engineering.
- Thinking about appropriate problem solving strategies: thinking of appropriate algorithms, data structures, modeling and programming techniques.
- Expressing and communicating the solution: writing the design document, implementing the solution as a computer program, documenting the program.
- Checking the solution: running, debugging, and testing the program, designing appropriate test cases, reasoning about correctness of the program.

Suggestions for Teaching
- Provide students with an interesting setting in which they would experience engineering, problem solving, and issues related to Software Engineering.
- Provide hands-on experience through thought provoking examples of formulating a problem, designing a solution, checking the solution, and communicating the solution.
- Discuss programming techniques and their applicability to effective problem solving.
- Teach students to read, write and reason about programs.
- Discuss, understand, and practice how one learns new material.
- Provide a platform for the students to understand the major items and changes that are required from them to learn in a university setting. One of the purposes of this class is to talk to the students about the major changes and adopting capabilities they need to develop for a successful learning experience in a university setting.
- Provide students opportunities to practice problem solving in a group setting. Guide them to work effectively as a group.
- Make students aware of current technologies and tools and their effective use for problem solving (e.g. use of an Integrated Development Environment such as Eclipse, object-oriented programming, modeling methodology such as UML, debugging tools etc.).
- Overall, guide and encourage students to think and ask good questions.
**SE 309: Software Development Practices**

**Catalog Description**

SE 309: Software Development Practices

(3-1) Cr. 3 F.S. Prereq: SE 185, Com S 228, Engl 104. A practical introduction to methods for managing software development. Process models, requirements analysis, structured and object-oriented design, coding, testing, maintenance, cost and schedule estimation, metrics. Programming projects. Nonmajor graduate credit.

**Learning objectives**

The main objective is that students understand (and experience) that coding is only a small part of the overall activities towards a software development project. An important objective is that students understand that unless errors are trapped during the upstream activities (requirements, design) – the project objectives of being on schedule, budget, and quality are in jeopardy. Specifically, at the end of the semester, students will:

- Learn the steps, techniques, issues, and tools of the engineering processes: requirements, architecture, coding, inspections and testing.
- Learn the steps, techniques, issues, and tools of the management processes: planning, scheduling, cost estimation, risk management, change management, and configuration and release management.
- Experience working as part of a team to build a large software project – starting from conception to release.
- Practice developing key artifacts such as requirements document, architecture document, test-plan, software development plan, inspection reports, and end-of-project report.
- Learn the basics of Capability Maturity Model and what is involved in process improvement. Also, learn about important process metrics and about IEEE standards.
- Develop skills in using some standard software tools such as: build (make), source control (CVS), modeling (Rational Rose), change/issue tracking (RT), test-manager, Integrated Development Environment (IDE) (Visual C++), project management (Microsoft Project), etc.

**Sequence of teaching modules**

1. Introduction to Software Engineering (Chapter 1; 2 lectures)
2. Software processes and management (Chapters 3 & 4; 23 & 24; 2 lectures)
3. Software Requirements Specifications (Chapters 2 & 5-9; 5 lectures and in-class exercise)
4. Software Design (Chapters 10-15; 8 lectures)
5. Critical Systems (Chapter 16; 2 lectures)
6. Verification and Validation (Chapter 19-20; 3 lectures and in-class exercises: Test plan development)
7. Maintenance and evolution (Chapter 27; 2 lectures: Final Exam during exam week)
**Suggested Textbook**

**Reference Books**
3. UML Toolkit, by Hans Eriksson and Magnus Penker.

**Labs and Assignments**
Labs and Assignments geared towards development of skills in using some standard software tools such as: build (make), source control (CVS), modeling (Rational Rose), change/issue tracking (RT), test-manager, IDE (Visual C++), project management (Microsoft Project), etc.

**Project**
Medium sized project to help students experience working as part of a team to build a large software project – starting from conception to release and to practice developing key artifacts such as UI designs, requirements document, architecture document, test-plan, software development plan, inspection reports, and end-of-project report.

**Additional teaching considerations**
- If possible, have students in SE 329 (Software Project Management) manage the project for this class. In this way, the students in this class will have an experienced student guide their project – whereas the students of SE 329 will be able to practice planning and tracking and helping projects to finish on time.
- Projects should stretch the student’s abilities and have them encounter real difficulties. For example, if students were to learn a new language, or a new tools and technologies (such as Web based technologies).
- Projects must enforce use of an IDE, a source control tool, a project management tool, and testing tools.
- Project presentations of the Best Projects should be announced to the entire department. This will allow comparisons with the capstone project presentations.
SE 319: Software Construction and User Interfaces

Catalog Description


Learning objectives

- Be able to apply a wide variety of software construction techniques and tools.
- Learn the concepts of software modeling.
- Learn the role of prototyping in project development.
- Learn UML and UML tools.
- Learn user centered design and usability engineering principles as they design a wide variety of software user interfaces.
- Practice requirement analysis, design, and testing skills by applying them to programming assignments.

Suggested sequence of teaching module

1. Construction technologies and tools
   a. Grammar-based input processing, parsing
   b. Code re-use, libraries
   c. Error handling, exception handling
   d. Tools: scripting, markups, macros etc.
2. System modeling (Sommerville 7, Hamlet 14,17)
   a. State machines, Data-flow diagrams, Object-oriented models
   b. Basics of modeling discrete/continuous systems and simulation
3. Object-oriented modeling and UML (Sommerville, Bruegge 6,7,8,9)
   a. Decomposition
   b. Re-use
   c. Interface specification
4. Software prototyping (Sommerville 8 and online reference materials)
   a. Evolutionary/throw-away prototyping
   b. Domain dependent prototyping languages for
      i. Interactive systems
      ii. Database applications
      iii. Symbolic processing
5. User interface (Sommerville 15, Pressman 12,)
   a. Types of user-interface
   b. UI design
   c. Overview of tools for developing UI
6. Domain specific software engineering
   a. Web engineering
   b. Safe critical system
c. Embedded systems, etc.

**Suggested Textbook**


**Reference Books**


**Labs and Assignments**

- Programming assignments to cover various construction techniques.
- Exercises building models in UML, particularly class diagrams and state machines.
- Use of software engineering tools to create designs.
- Paper prototyping of user interfaces, then discussing design options in order to arrive at a consensus design.
- Implementation of a system with a significant user interface component using a rapid prototyping environment.

**Project**

- Writing an expository paper to explain what is model-based software development and with illustrative examples
- Writing an expository paper on domain-specific software development selecting a specific topic such as real-time systems, embedded systems, parallel programming or an application area such as bio-informatics.

**Additional teaching considerations**

- Students should be asked to write requirements, design documents, test plans for the programming assignments. The objective should be to get students in the habit of thinking about these issues and documenting the main points.
- Students should be taught to obtain informed consent from users when involving them in the evaluation of user interfaces.
- Small groups of students present paper prototypes of their UI designs to the class. Other students in the class then express what they like about the designs. Next, the other students provide constructive criticism.
- Introduce students to the project topics early in the semester and give them a good introductory paper or book chapter to read. Make it clear how they should expand on the material contained in the reading material you have given. Divide the project into a series of submissions such as literature survey, first draft, and second draft. The objective is to ensure that students do spend time on it over an extended period and iteratively improve their paper. You may also want to ask each student to read two drafts prepared by their fellow students and provide constructive criticism.
SE 329: Software Project Management

This will be the third course in a sequence of four required software engineering courses.

Catalog Description


Learning objectives

Upon completion of this course, students will have the ability to:

- Understand process-based software development
- Develop a comprehensive project plan for a significant development effort
- Apply management techniques to projects that follow agile methodologies, as well as methodologies involve larger-scale iterations or releases
- Effectively estimate costs for a project using several different techniques.
- Apply function point measurement techniques
- Measure project progress, productivity and other aspects of the software process
- Perform risk management, dynamically adjusting project plans
- Use configuration management tools effectively, and apply change management processes properly
- Draft and evaluate basic use of standards in project management, including ISO 10006 (project management quality) and ISO 12207 (software development process) along with the SEI’s CMM model
- Assess a software process to evaluate how effective it is at promoting quality
- Conduct effective and efficient inspections
- Manage test plans and assess their coverage

Suggested sequence of teaching module

1. Process-based approach for project execution
2. Capability Maturity Model for Software
3. Proposals and contracts
4. Requirement specification and management
5. Process definition and tailoring
6. Process database and process capability baseline
7. Effort estimation and scheduling
8. Risk management
9. Project management plan
10. Configuration management
11. Life cycle execution
12. Project monitoring and control
13. Project audits
14. Project closure
15. Quality planning and quality metrics
16. Applying basic quality tools in software development
17. Code reviews
18. Software testing
19. Software metrics

Suggested Textbook


Reference Book


Labs and Assignments

- Compare and contrast two software processes
- Write a survey article on project management tools.
- Make cost estimates for a small system using a variety of technique
- Writing a configuration management plan
- Using change control and configuration management tools
- Evaluating a software contract or license
- Inspecting of software in teams; comparison and analysis of results
- Compute test coverage and yield, according to a variety of criteria

Project

Students will develop a project plan for building a project management system. This will be a half-semester project. The industry case study to be covered in class will serve as an example from where students will draw ideas for creating a substantial project plan. In the software architecture and design course, the students will build the project management system.
SE 339 Software Architecture and Design

This will be the last course in a sequence of four required software engineering courses.

Catalog Description

Learning objectives
- Understand why the architectural design of software is important
- Learn about different architectural styles
- Be able to do architectural designs, analysis and evaluation
- Be able to apply design patterns
- Use sound quality metrics as design objectives and assess the designs to ensure the objectives have been met
- Learn to design to meet requirements such as safety and security
- Use reverse engineering techniques to recapture the design of software

Suggested sequence of teaching module
1. Architectural design decisions
2. Architectural system models
3. Criteria for good software architectures
4. Analysis and evaluation of architectures
5. Iterative development process
6. Methodical approach to OO design
7. Design patterns
8. Domain-specific constraints and frameworks
9. Software complexity and metrics
10. Distributed system software
11. Reverse engineering
12. Architecture definition languages

Suggested Textbook
Reference Books: (some material will be taken from the following references)


Labs and Assignments

- Develop a DTD for a simple architecture description language. Test your DTD by writing an XML document instance that exercises all features of your DTD.
- Architecture design assignments covering different architecture styles.
- Assignment to do architectural analysis and evaluation
- Assignments to practice the use of design patterns
- Assignment to extract architecture by reverse engineering

Project

The project will like a trial run before the capstone project. Students will practice important software engineering activities such as:

- Go through the software development life cycle starting from architectural design to implementation.
- Follow the iterative development process.
- Write documents as an integral parts of the development process.

Additional teaching considerations

- A project in this course could be to develop a tool for project management. In subsequent years students can be asked to enhance the tool developed during the past year. Eventually a sophisticated tool can be developed through the course. This will also force the students to revisit the project management process and think about it very carefully.
- Students can be asked to go through some non-trivial code and extract its architecture. Reading and understanding code written by others is a very important activity in real-life software engineering.
- As a part of the project each student team can be asked to review architectural designs and other documents of two other teams so provide constructive criticism in written form. This will foster collective learning.
SE 409: Software Requirements Engineering

Catalog Description
SE 409. Software Requirements Engineering
Cr. 3. Prereq: SE 309, SE 319. The requirements engineering process, including identification of stakeholders, requirements elicitation techniques such as interviews and prototyping, analysis fundamentals, requirements specification, and validation. Use of Models: State-oriented, Function-oriented, and Object-oriented. Documentation for Software Requirements. Informal, semi-formal, and formal representations. Structural, informational, and behavioral requirements. Non-functional requirements. Use of requirements repositories to manage and track requirements through the life cycle. Case studies, software projects, written reports, and oral presentations will be required.

Learning objectives
Upon completion of this course, students will be able to:

- Plan and implement a requirements engineering effort,
- Understand and identify problems and approaches in capturing requirements,
- Understand fundamentals of analysis,
- Be aware of and be able to represent requirements in form of specification documents, and
- Know how to use tools and techniques for managing large requirements repositories.

Suggested sequence of teaching modules
1. Need for requirements specifications.
2. Requirements engineering process overview
3. Identification of stakeholders.
4. Requirements elicitation and capture techniques including prototyping.
5. Modeling: state-oriented, function-oriented, and object-oriented.
7. Types of Requirements: structural, informational, behavioral, non-functional, and domain.
8. Informal, semi-formal, and formal representation of requirements. Types of requirements artifacts.
9. Validation of requirements (Inspections).
10. Use of requirements repositories to manage and track requirements through the life cycle.
11. Case studies and projects (including written reports and oral presentations).

The course will be taught using a set of papers and selected material from the following reference books.

Reference Books:

**Labs and Assignments**

Labs and assignments geared towards development of skills in using Requirements Repositories and development of Elicitation and Analysis skills.

**Project**

Medium-sized project where students plan and implement a requirements engineering effort. They will apply the principles and techniques discussed in lecture to the problems of eliciting, documenting, and validating the specification of a non-trivial software system.
SE 412: Formal Aspects of Specification and Verification

Catalog Description

Learning objectives
Upon completion of this course, students will have the ability to:

- Understand fundamentals of logic and be able to specify program correctness properties using predicate and temporal logic.
- Understand the formal aspects of theorem proving and how to use a theorem prover. Be able to specify model and axioms and theorems using a theorem prover and then be able to prove theorems using the prover. Understand the use of at least one theorem prover.
- Understand the formal aspects of model-checking and how to use a model-checker. Be able to create models, specify properties, and then use a model checker to verify these properties. Understand the use of at least one model checker.
- Be able to apply knowledge to a small application. In particular, they should identify different aspects of the problem that can benefit from formal techniques, select appropriate techniques, and then apply these techniques.

Current sequence of teaching modules
2. Predicate logic, Deduction/Inferencing in predicate logic.
3. Specifying program for correctness.
4. Hoare triples and program verification by weakest precondition method.
5. Partial and total correctness of programs.
6. Temporal logics: CTL, CTL*, LTL
7. Model checking, symbolic/bounded
8. Program verification using model-checking: Slicing, Abstraction, Counterexample
9. Case studies and projects (Written and Oral presentations).

Suggested Textbook
Logic and Language Models for Computer Science, by Henry Hamburger and Dana Richards, ISBN 0130654876

Reference Book:

Labs and Assignments
- Prove propositional and predicate logic formulae using inference rules.
- Specify properties of program segments using predicate logic.
- Specify models and prove theorems using a theorem prover.
- Specify and manipulate expressions in temporal logic.
- Specify models of programs in a model checking language and specify their properties. Verify properties using a model checker.

**Project**

- Work as a team of 2-3 on a project involving formal modeling and verification techniques.

- Students will select a small application in consultation with the instructor. They have to identify aspects of the problem that can benefit from formal modeling and verification techniques, select appropriate techniques for the different aspects, apply the formal techniques and obtain modeling and verification results.

- Teams will have to write a report and give an oral presentation of their results.
SE 416: Software Evolution and Maintenance

Catalog Description

SE 416. Software Evolution and Maintenance Cr. 3. Prereq: SE 309, SE 319 Fundamental concepts in software evolution and maintenance; practical software evolution processes; legacy systems, program comprehension, impact analysis, program migration and transformation, refactoring. Tools for software evolution and maintenance. Case studies, experimental software projects. Written reports and oral presentations. Nonmajor graduate credit.

Learning objectives

Upon completion of this course, students will have the ability to:

- Understand challenges of evolving and maintaining software over long periods of time.
- Understand the evolution process, the possible alternatives and their tradeoffs.
- Apply different methods, techniques, and tools to facilitate software evolution and maintenance.
- Work with legacy systems.
- Understand implications of design and development decisions to long-term maintenance and evolution of software.

Suggested sequence of teaching modules

1. Real-life examples of evolution: Linux, Windows, Eclipse
2. Different types of evolution scenarios: open source software, proprietary software, software platforms
3. Different types of maintenance activities: corrective, adaptive, perfective, and preventive
4. Estimating maintenance costs and productivity
5. Predicting maintainability with software quality metrics.
6. Legacy systems and evolution and maintenance costs
7. Program understanding at different levels of abstraction
8. Different approaches to program understanding: documentation, static and dynamic analysis
9. Program slicing methods
10. Software visualization techniques
11. Program understanding tools
12. Reverse engineering: extracting architectures and patterns from source code
13. Software reengineering activities: migration to different platform, changing underlying logic, reorganizing software, interface reengineering, and performance optimization.
14. Refactoring software
15. Source-level program transformations
16. Designing maintainable systems – aspect-oriented software design and other paradigms
17. XML as software integration technology
18. Impact analysis

Reference Books: (Set of papers and selected material will be used from these references)


Labs and Assignments

- Extract software architecture artifacts from code and documentation and write a report that illustrates the architecture of a selected UNIX utility.

- Write a survey report on legacy software in financial services industry and the available technologies and tools for maintaining and evolving those systems.

- Identify different features provided by various program analysis tools. For each feature, its purpose, the type of underlying program analysis, evaluate its usefulness, and suggest enhancements.

- Identify different software visualization paradigms. Compare and contrast their applicability by considering different scenarios for program understanding.

- Read about different program transformation techniques. Describe the commonality and differences between different techniques. Suggest a scheme to classify various transformation techniques.

- A number of reengineering studies are described in some of the reference books and other places in the literature. Read one study where you think the authors have provided comprehensive information. Describe different aspects covered in the study and summarize it. Write a self-contained report that a software engineering student will find interesting and informative.

- Describe the applicability and usefulness of aspect-oriented programming. Illustrate it with a concrete example of a program written by you using the AspectJ tool.

- Describe the differences between different impact analysis techniques.

- Read about Eclipse platform and describe its plug-in architecture. Experiment with a plug-in of your choice and describe your experience. Describe what and how you can refactor Java code using Eclipse.

- Describe the XML technology and its use in software integration.

- Investigate and describe in detail how an application can be made end-user programmable and extensible using a scripting language (e.g., Perl, Javascript, Tcl).
SE 417: Software Testing

Catalog Description
SE 417. Software Testing Cr 3. S. Prereq: SE 309, SE 319. Comprehensive study of software testing, principles, methodologies, management strategies and techniques. Test models, test design techniques (black-box and white-box testing techniques), integration, regression, system testing methods, and software testing tools.

Learning objectives
- Have an ability to apply software testing knowledge and engineering methods.
- Have an ability to design and conduct a software test process for a software testing project.
- Have an ability to identify the needs of software test automation, and define and develop a test tool to support test automation.
- Have an ability understand and identify various software testing problems, and solve these problems by designing and selecting software test models, criteria, strategies, and methods.
- Have basic understanding and knowledge of contemporary issues in software testing, such as component-based software testing problems.
- Have an ability to use software testing methods and modern software testing tools for their testing projects.

Sequence of teaching module
- Introduction to software testing
- Qualities and fault models
- Software testability
- Black-box testing
- White-box testing
- Integration testing
- Coverage and Regression testing
- System testing and performance measurement
- Software test automation and tools
- Software quality assurance
- Domain specific issues
  1. Interprocedural
  2. Pointer
  3. Object oriented
  4. Real time
  5. Component based
- Evaluation of testing: Formal/Experimental
Suggested Textbook


Reference Books


Projects

1. Plan, design, and test software components or system based on a well-defined test coverage criteria.
2. Learn and study advanced topics, such as component-based software testing, and test automation.
Appendix C: Comparison with Computer Engineering Degree

Comparison of Software Engineering Program Requirements with College of Engineering and Computer Engineering Program Requirements

This describes how the proposed undergraduate Software Engineering (SE) program meets the basic program requirements in the College of Engineering. Also, it points out differences from the Computer Engineering (CprE) program requirements.

1. Comparison with the Basic Program in Engineering Requirements (25.5 credits)

The courses that constitute the Basic Program in Engineering are listed at the website given below. All engineering students are required to complete the Basic Program. Freshman may have a one-semester overlap between their Basic Program coursework and 200-level engineering courses. Transfer students may have a two-semester overlap. The average grade for the courses in the Basic Program must be C- or higher.

Reference: http://www.eng.iastate.edu/eup/tier2pages/course_information/supportpages/basic.asp

<table>
<thead>
<tr>
<th>Engineering Requirement</th>
<th>Credits</th>
<th>SE Program</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemistry 167</td>
<td>4</td>
<td>Chemistry 167</td>
<td>4</td>
</tr>
<tr>
<td>Lib 160</td>
<td>0.5</td>
<td>Lib 160</td>
<td>0.5</td>
</tr>
<tr>
<td>CprE 185 (problem solving)</td>
<td>3</td>
<td>SE 185 (problem solving)</td>
<td>4</td>
</tr>
<tr>
<td>Engineering 101</td>
<td>R</td>
<td>Engineering 101</td>
<td>R</td>
</tr>
<tr>
<td>Math 165, Math 166</td>
<td>8</td>
<td>Math 165, Math 166</td>
<td>8</td>
</tr>
<tr>
<td>Physics 221</td>
<td>5</td>
<td>Physics 221</td>
<td>5</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>26.5</strong></td>
<td></td>
<td><strong>27.5</strong></td>
</tr>
</tbody>
</table>

- The Chemistry 167 can be substituted by a biology course if such a course is approved. Given the emerging importance of Bioinformatics, a biology courses will be very appropriate.

2. Comparison with the Computer Engineering Program Requirements

The following table compares the current Computer Engineering and the proposed Software Engineering programs.

---

1 Refer to pages 11-12 for the complete description of the software engineering curriculum.
<table>
<thead>
<tr>
<th>Computer Engineering</th>
<th>Credits</th>
<th>Software Engineering</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering College Requirements:</td>
<td>26.5</td>
<td>Engineering College Requirements:</td>
<td>27.5</td>
</tr>
<tr>
<td>Communication etc. English 314 (3)</td>
<td>3</td>
<td>Communication etc English 314 (3), SpComm 212 (3), Econ 101 or Econ (102) or IE 305 (3)</td>
<td>9</td>
</tr>
<tr>
<td>Seminars: CprE 166, CprE 494</td>
<td>R</td>
<td>Seminars: CprE 166, CprE 494</td>
<td>R</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>124.5</strong></td>
<td><strong>TOTAL</strong></td>
<td><strong>124.5</strong></td>
</tr>
</tbody>
</table>

2 The total number of credits remains fixed at 124.5 as the students who choose the higher credit option courses CprE 211 and CprE 308 can take up to two less credit hours of electives.
Appendix D: Comparison with the Computer Science Degree Program\textsuperscript{3}

<table>
<thead>
<tr>
<th>Computer Science Program</th>
<th>Credits</th>
<th>Software Engineering Program</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Course Requirements:</td>
<td>45</td>
<td>SE Introductory courses:</td>
<td>50/48</td>
</tr>
<tr>
<td>ComS104, ComS 227, ComS 228, CprE 210, SE 309, ComS 311, ComS 321, ComS 330, ComS 331, ComS 342, ComS 352, ComS 362/ComS 363,</td>
<td></td>
<td>ComS 227, ComS 228, ComS 229/CprE 211, CprE 210,</td>
<td></td>
</tr>
<tr>
<td>Additional courses:</td>
<td></td>
<td>SE Intermediate courses:</td>
<td></td>
</tr>
<tr>
<td>Two from the following groups with at least one from Group (2)</td>
<td></td>
<td>ComS 311, ComS 352/CprE 308, ComS 363,</td>
<td></td>
</tr>
<tr>
<td>(1) ComS 411, 440, 454, 472</td>
<td></td>
<td>SE core courses:</td>
<td></td>
</tr>
<tr>
<td>(2) ComS 401, 425, 430, 461, 472, 474, 476</td>
<td></td>
<td>SE 185, SE 309, SE 319, SE 329, SE 339, SE 441, SE 442</td>
<td></td>
</tr>
<tr>
<td>(3) ComS/Math 471, Math 481, ComS 485,489, ME 557</td>
<td></td>
<td>SE Electives:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Two from Software Engineering Electives:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software Requirements (409)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software Verification (412)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software Evolution (416)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software Testing (417)</td>
<td></td>
</tr>
</tbody>
</table>

| LAS Requirements:\textsuperscript{4} | 80.5 | Complimentary Courses and Electives: | 75.5/77.5 |
| Electives | | Total | 125.5 |
| 14 Credits of Electives | | Total | 125.5\textsuperscript{5} |

\textsuperscript{3} Refer to pages 11-12 for the complete description of the software engineering curriculum.

\textsuperscript{4} Computer Science Majors are required to have 1 yr college level or 3 yr high school level foreign language (see http://www.cs.iastate.edu/ugrad-info/catalog5.html)

\textsuperscript{5} The total number of credits remains fixed at 124.5 because the students who choose the higher credit options CprE 211 and CprE 308 can take up to two less credit hours of electives.