Software Engineering Course Group Report

Date of report: Sep. 7, 09

Software Engineering Group

Course no.	Title	Credit Hours	Reqd - Core (R)/ Reqd - Option (O)/ Elective (E)	Capstone?
CSE 502	Object-Oriented Programming for Engineers and Scientists	3	E	
CSE 560	System Software Design, Development, and Documentation	5	R	
CSE 601	Social and Ethical Issues in Computing	1	R	
CSE/ECE 668	Applied Component-Oriented Programming for Engineers and Scientists	3	E	
CSE 757	Software Engineering	3	O	
CSE 758	Software Engineering Project	4	E	Yes
CSE/ECE 767	Applied Use-Case-Driven Object-Oriented Analysis and Design for Engineers and Scientists	3	E	
CSE/ECE 769	Applied Enterprise Distributed Computing for Engineers and Scientists	3	E	
CSE 794J	Enterprise Architecture I	3	E	
CSE 794K	Enterprise Architecture II	3	E	

1. Summary

Software engineering is a core area in the discipline of computer science. Over the last decade, software and information technology have been primary drivers of economic growth in the US. Developing reliable software, however, remains a challenge. A recent NIST report (June 2002) estimates that software errors cost the US economy 59.5 billion dollars annually. Also, the practice of software engineering is evolving, with the emergence of agile practices, the need to consider technology management, legacy software integration, organizational management, as well as deployment and infrastructure issues instead of a focus on just developing software. The software engineering course group plays a pivotal role in addressing this challenge by teaching good engineering practices, and in developing the appropriate professional and social ethics.

The software engineering course group consists of two sets of courses:

1. *Required and primary elective courses, or the "core" sequence* - CSE 560, CSE 601, CSE 757, and CSE 758: These courses are intended for computer science majors and are either required for the major (560 and 601), required for the software systems option (757), or strongly encouraged (758).
2. *The "Applied Software Engineering"* sequence - CSE 502, CSE/ECE 668, CSE/ECE 767, and CSE/ECE 769: These courses are intended for students from other science and engineering majors, but with an interest in state-of-the-practice software engineering principles. They form a cohesive curricular unit, with CSE 502 serving as the prerequisite to CSE/ECE 668, and CSE/ECE 668 serving as a prerequisite to both CSE/ECE 767 and CSE/ECE 769. It is important to recognize that all these courses except CSE 502 are appropriate for CSE/CIS majors to enhance their software engineering backgrounds.
3. A new pilot course sequence (CSE 794J and CSE 794K) on Enterprise Architecture.

The first course in the core sequence is CSE 560 ("System Software"). CSE 560 is the keystone course of the CSE curriculum: it is a required course for all CSE students and it serves as a prerequisite for the majority of upper-level courses. The technical content of this course centers on the tools (e.g., compilers, assemblers, linkers, loaders, operating systems) that bridge the semantic gap between the machine-level view of instruction execution and the applications developer view of programs. This course expands significantly on several topics that were first introduced in the software spine: teamwork, design, and technical writing.

Required in this group is CSE 601 ("Social and Ethical Issues in Computing"), a required course on the social, ethical, and legal issues that arise in the practice of the computing profession as well as in the application of computer technology. This course is structured around student presentations and debates on current issues in the field such as privacy rights, security rights, copyright laws, and antitrust laws. Emphasis is placed on oral communication and on the ability to hold and defend reasoned opinions. This course is a prerequisite for all Capstone courses, and hence for CSE 758 (see below).

CSE 560 is followed by CSE 757 ("Software Engineering") where principles of requirements, analysis, architecture, design, implementation, validation, deployment and management of computer software are taught. Over the last year, an emphasis on "enterprise" software engineering is being incorporated in the course.

The final course in the core sequence is CSE 758 ("Software Engineering Project") that fulfills CSE Capstone requirements. The major objective of this course is to ready students for the marketplace by exercising--through a significant, large-scale project--the knowledge they have gained in previous software and software engineering courses.

Course projects typically involve practical (real) problems for a larger client community. Specifically, upon completing this course, students will have applied, in a real project, with a real customer, their knowledge about the software development lifecycle (SDLC), including requirements, analysis, architecture and design, project management, implementation, testing and deployment using a range of software development tools and technologies.

In the Applied Software Engineering sequence, the first course, CSE 502, has largely traditional content. It is required to get engineering and science students with non-object-oriented-computing backgrounds “up to speed” in using an object-oriented language to develop programs. It focuses on standard introductory object-oriented concepts and programming language features that are routinely taught to undergraduate computing majors. Students who have this background already may skip this course.

The next three courses, CSE/ECE 668, CSE/ECE 767, and CSE/ECE 769, teach and apply software engineering processes based on modern industry-standard technologies such as UML; iterative and incremental development; use-case analysis; component-based software engineering; Java technologies including Java Beans, reflection, and class loading; C# and .NET technologies from Microsoft; and distributed systems based on standards such as EJB, JSP, XML, SOAP Web Services, WS-* specifications, the Enterprise Service Bus (ESB) and “cloud” computing.

The pilot 2-course sequence CSE 794J and K is on Enterprise Architecture, i.e. how complex organizations are structured and how technology supports the business. This course should ideally fit in the Applied Software Engineering sequence, as an important course for computer science graduates seeking to enter professional practice, because of the relevance of organizational design and management, technology management, project and program management and vendor management to professionals.

Finally, 2 other courses – CSE 421 Software Development in Java, and CSE 616 – Object-Oriented Systems Analysis are being considered for incorporation into this course group. This report will be extended to include this courses when this happens.

Overall, students and faculty are generally satisfied with the courses in the software course group. CSE 560 remains one of the most challenging and intense courses in the curriculum, but it is also perceived by the students as one of the most relevant for their future professional careers. The initial offerings from the Applied Software Engineering course sequence have been well-received, although enrollment from outside the department has not been as robust as we might like. The Enterprise Architecture course is primarily taken by graduate students.

2. Detailed Analysis

Section 2.1 describes the individual courses in the group. Section 2.3 explains how the group is related to the rest of program. Section 2.4 explains how the group helps achieve a number of CSE Program Outcomes. Section 2.5 describes the feedback we have received from students, recruiters, etc., about the courses in the group. Section 2.6 summarizes the main changes we have made in the courses since the previous report. Section 2.7 discusses some continuing and emerging concerns involving these courses and our plans to address them.

2.1 Summary of the courses

CSE 560: This is a five-credit course that serves as the keystone of the curriculum. It is a prerequisite to almost all upper-level courses and students generally take this course during their sophomore or junior year. The class meets for 4 lecture hours/week. The extra credits reflect the heavy lab component of the course and the need for periodic design review meetings with the instructor and grader.

The technical content of the course centers on system software (e.g., compilers, linkers, loaders, interpreters). The emphasis of the course, however, is as much on design, documentation, implementation, testing, teamwork, and communication skills as it is on the pragmatics of system software.

Over the course of the quarter, students design, document, implement, and test several projects. These are large programming assignments, some taking as many as 6 weeks to complete, depending on the instructor. Students are given a description of the abstract machine for which they are building the system software and a good description of the required functionality for each particular assignment (e.g., an assembler). Like many real-world engineering situations, however, the assignment specifications are somewhat incomplete and occasionally ambiguous. It is up to the students to address the omissions and ambiguities by soliciting information from their hypothetical client and by making appropriate design choices. Apart from design choices in their system requirements, students are also free to make many design choices in the completion of their assignments; for example, the implementation language and platform as well as the module decomposition, data structures, and algorithms. For most students, CSE 560 is the first time they will face ambiguous requirement requests and the first time they will be given significant latitude in how they satisfy these requirements.

CSE 560 is also the first time most students will be building programs that are too large to complete in a single sitting. At the beginning of the quarter, the class is divided into teams of four or five. Assignments are completed by the team as a whole, and the entire team is given a single grade (with the exception, in some sections, of 5% that is reserved for individual effort and is allocated according to a peer review by other group members). Teams are responsible for setting up regular meeting times and for partitioning the workload equitably. Teams meet with the instructor during the quarter to get direct feedback on their design decisions as well as to analyze and discuss how well the team is working together. Peer evaluations are submitted regularly during the quarter and provide more insight into the team dynamics that have evolved. Students gain first-hand experience in the challenges involved with working with others. They learn the importance of proper task allocation, how to rely on others, and how to take responsibility for completing tasks both as individuals and as a group.

In addition to a working implementation, teams must also deliver extensive documentation in the form of a user's guide, a programmer's guide, a testing guide, and the minutes from all team meetings. CSE 560 is officially designated as a third writing course. It is not uncommon for this documentation to exceed several hundred pages over the course of the term. Recently, some sections of the course have experimented with online and electronic documentation. Some lecture time is set aside to address technical writing issues. For the first assignment, draft documentation is submitted so that students can get feedback before submitting their first set of documentation. Also, the last assignment is a synthesis of the previous assignments into a single working system and hence much of the documentation submitted for this final assignment is a rewritten form of previously submitted and graded documentation. Also, the College of Engineering maintains a Technical Communication Resource Center whose mission it is to "assist students in the development of effective written and visual communication skills". Students are encouraged to avail themselves of this service. There are two recommended textbooks for CSE 560, one of which is a reference book and style guide for technical writing.

CSE 601: The primary objective is for each participant to become informed, and to develop reasoned opinions about the ethical, social, and legal dimensions of various situations that may be encountered by computer professionals, and about social issues that arise from computer technology. Participants practice how to: hold an opinion, have reasons for one's opinions, change opinions when compelled by best judgment, and rationally discuss opinions in cooperative pursuit of truth and good practices. Additional objectives are to develop communication skills, both written and oral, and to encourage ethical and responsible behavior.

The class meets once a week, with a 90 minute time slot. Each student makes one 10 minute classroom presentation on an assigned topic -- either a report-style presentation or an assigned role in a debate. Debates follow a modified town-hall debate format with points of order and points of information. Each student also writes a three-page paper on a topic chosen from a list of possible topics.

The presentation grade is based on: effectiveness of communication, clarity, strength of preparation, accuracy of reported material. Students are given an opportunity to rewrite their papers based on instructor and grader feedback.

Changes to this course have been minimal, and are primarily due to the inclusion of newer current topics. The grading rubrics have been revised slightly based on experience. More communication is done through email. The overall external appearance of the course hasn't changed much, but the internal emphasis on professionalism is now clearer. For 2009, the requirement for student built web sites has been dropped due to privacy issues. The availability of GoogleDocs, blogs, presentation, and web sites is likely to change what the class should be trying to accomplish and what should be required.

For the switch to semesters, the instructor recommends keeping this a separate 1-hour course and adding some formalized case studies (the primary thing left out now). A desire of the instructor is to offer a version of this course for students in major other than computer science. Other engineering departments offer this kind of course, but majors from other colleges might also enjoy it.

CSE 757: This is our software engineering fundamentals course and is required for students in the Software Systems option. The course is a prerequisite for CSE 758 and CSE 772, and students generally take it during their junior or senior year. The class meets for three lecture hours per week.

Starting recently, this course is being extensively re-worked (with this re-work being supported by an NSF CCLI grant). The technical content of the reworked material now centers on software engineering techniques and methodologies related to the development and maintenance of large software systems, known as enterprise systems. The course studies the principles and methodological frameworks necessary to support the software development lifecycle (SDLC) of these enterprise systems. Topics include frameworks for understanding the business context of enterprise systems, requirements identification and analysis, project management (including project planning, estimation, configuration management and project risk management), software architecture and design, software quality assurance and testing, and software system deployment. There is a paper project where students apply the theory that is taught in the lecture. In Winter of 2009, in collaboration with Thoughtworks Inc., we developed an "organizational simulation"--a Lego^{TM} based game--that allowed students to experience some essential elements of Agile methodologies such as iteration planning, estimation, and the importance of customer engagement. This game was considered very effective by the students in showing how Agile works, and we will permanently incorporate this simulation into the course. We are working on developing simulations that illustrate other concepts.

The collaboration with Thoughtworks, Inc., is one example of industry engagement in the ongoing refinement of this course.

In the recent offerings of CSE 757, the course material has consisted of extracted material from a variety of books and papers, as well as exemplar SDLC artifacts from actual projects.

CSE 758: The major objective of this course is to ready students for the marketplace by requiring them to exercise the knowledge they have gained in previous software and software engineering courses on industry-specified real world problems. Specifically, upon completing this course, students are expected to have applied their knowledge of enterprise-scale software engineering frameworks to:

- The process of developing software:
    - Requirements and analysis
    - Software architecture and design, driven by non-functional requirements: scalability, security, usability and performance
    - Project management, planning, team structure, roles and responsibilities, Configuration management

    - Incremental and iterative, workbook-centered, object-oriented, structured and agile software development
    - Project planning
- Presenting work to an audience of peers
- Business issues in software development

Specific tools and techniques that students typically use include:

- Object oriented analysis, design and implementation work-products.
- Structured and agile methodologies.
- Unified Modeling Language (UML) notation for some of the above work products.
- Configuration management tools (RCS, SCCS, Ant, Make, or other), and other miscellaneous tools - Web authoring tools, Word processing tools etc.
- J2EE tools and components: Java 2 SDK, JSP pages and servlets, Enterprise Java Beans, JDBC, JUnit.
- .NET tools and components: SQL-Server, ASP.NET, C#.
- Development frameworks and toolkits for Mobile computing – such Windows CE, J2ME and proprietary frameworks
- Other frameworks, such as AJAX, Flex and Flash
- Relational databases
- Documentation/Markup languages including HTML, XML, and Javadoc.

Projects in this course have been many and varied. Projects are all provided from external sources--large and small industry, and individuals within and outside the university. All of them are sponsoring the project because of a real need. In all cases, the external entity has assigned one or more of their personnel to work with the students--mostly in the provision of requirements, but also as mentors. Finally, industry representatives attend the mid-term and final presentations, act as the "instructor-for-a-day" and provide feedback on how the students are proceeding, with specific feedback on what portions of the projects could make a suitable employment portfolio.

The number of projects done in a quarter depends on the number of students in a section. Some sections assign a project to sub-groups of 4-5 students. Other sections have typically assigned a single project to an entire section.

Recent projects have included e-commerce sites, mobile applications, a reading assistant for young children, a web-based geographical game, a web-site for microloans to people in the developing countries, and the re-architecture of an enterprise scale client-server system (sponsored by local entrepreneurs). Almost 50% of the projects have been taken into production by the sponsors later.

The use of real external projects, in almost all cases, has brought in significant and interesting complexity to the projects--much more so than if the instructor had formulated projects. Students have been very motivated because of the presence of a real customer. Some projects have been very successful, and are confidently showcased by both students and the instructors to external industry. Business requirements have been well-understood by students, and there is a clear connection to business and market requirements in the scope and design of the projects.

Each of these projects has required system and software design. The non-functional (design) requirements that students have concentrated on have primarily been UI requirements. An extremely high-performing group doing a referee-enrollment web site project considered scalability as a key design requirement, studying several design alternatives, and implementing connection pooling, caching, very lightweight templates and database optimizations. Another group (that did a SmartCard digital cash project) made security their primary design requirement, and brought in the use of considerable digital-cash technology. Students are required to make weekly presentations during the quarter to showcase their progress and solicit feedback and guidance. Students are also required to do two formal presentations--a mid-term one (45 minutes to 1 hour in length) and a final one lasting 2 hours. The mid-term presentation is made to the entire class, the instructor and invited external evaluators. The final presentation is made only to the evaluator and the instructor because of the difficulty of scheduling times when the entire class can attend. Each student individually presents at least 6 times during a quarter. Students clearly benefit from this presentation experience, as evidenced by the improvement that they make over the course of the quarter.

Even though the purported goal of the course is for students to apply their prior academic knowledge to class projects, students have not typically come in with a great deal of directly applicable prior knowledge. This means that students have to learn new things, such as new technologies and methodologies as well as learn to develop skills for applying their prior knowledge (on their own, and with mentoring, but without formal lecture from instructors). We believe that this experience will set the stage for life-long learning on the part of students--both in terms of expectations as well as ability.

There have been two offerings of CSE 758 with students who have taken the changed CSE 757 course. We believe (albeit anecdotally) that the student groups with a member who has taken the new CSE 757 perform better with respect to the process aspects of software engineering.

We were of the opinion that a two-quarter Capstone sequence was necessary for course expectations to "sink in", and be met in a better manner. However, with the course now having been incrementally streamlined, we no longer believe that this is necessary. The move to semesters will certainly help.

CSE 502: This is a three-credit course that introduces non-CS majors to object-oriented programming. It is intended for science and engineering students (lab assignments are technical in this direction), and serves as the transition course for non-CS majors who wish to take the Applied Software Engineering sequence. Topics include object-oriented programming concepts, interfaces and classes, and some important and standard design patterns. This is a course for either graduate or undergraduate credit. It is oriented primarily to graduate students in other departments who want an advanced introduction to basic principles of object-oriented programming and the use of a modern program development environment (Java and Eclipse). Even though there are minimal prerequisites, we expect the students to have experience with programming.

CSE/ECE 668: This course covers the application of component-based software engineering technology to design and implementation of software systems in engineering and science. The topics include:

- Review of basics of object orientation
- UML class diagrams and interaction diagrams
- Building efficient object-oriented software components
- Events and the Java Event Model as the basis for "wiring together" software components into a larger software system
- Introduction to JavaBeans
- Multithreading and its importance for correctness of engineering and scientific software systems
- Building a sample engineering and/or scientific software system

CSE/ECE 767: This course is focused on developing a software application sase study using incremental and iterative use-case-driven process of building object-oriented scientific and engineering software systems; analysis, design, UML modeling, design patterns.

Topics in this course include:

- Review of elementary UML and intermediate concepts
- What is good software?
- Introduction to the case study problem
- Requirements capture, and use case analysis
- Introduction to use case driven iterative and incremental processes
- Analysis and problem domain models
- Robustness diagrams
- Bridging the gap between analysis and design; GRASP patterns.
- Design and design patterns

CSE/ECE 769: This is a course introducing current technologies and middleware for distributed enterprise computing. Technologies include XML, JavaScript, Flash, Ajax, Flex, Enterprise Java, Web Services, an Enterprise Service Bus and the Google Web Toolkit. The objective of this course is to learn about current technologies in distributed enterprise computing and apply them in small projects in a disciplined manner. The class is organized into teams of two people each. The same projects are assigned to each team. Teams must develop and apply an appropriate software design methodology to the project. They are expected to meet outside the usual class periods for the project and maintain and present the project workbook in electronic form on the Web. Maintenance of the work-products on the web site is part of the class project, and their work-products on the web site are electronically archived at the end of the course.

The projects are intended to be small application components that build on each other to finally create a complete application. For the most recent offering of this course, the application was an emergency response application where crises and responders could be tracked on a map.

CSE 794J and K: This is a 2-course project-based sequence that introduces Enterprise Architecture and technology management. Topics include: interdisciplinary frameworks for enterprise Information Technology (IT) architectures; applications to transformation and innovation within the enterprise and creating solutions to real-world problems. Topics in 794J include:

- Enterprise Services and Architectures: trends, modeling and conceptualizing of macro-to-micro linkages, concepts of enterprise services composition and workflow integration.
- Overview of common architecture-related frameworks - ITIL (IT Infrastructure Libraries), TOGAF (The Open Group Architectural Framework), Architecture Tradeoff Analysis Method (ATAM), and Component Business Modeling.
- Introduction to business processes, supply chains, and enterprise systems.
- Adaptive Complex Enterprise: performance linkages between in-the-large and in-the-small, life-cycle and continuous improvement concepts, performance analysis and portfolio development.
- Introduction to policy formulation, implementation and evaluation. Discussion of traceability to meet regulations and requirements (such as Sarbanes-Oxley, HIPA and other security related) and their impact on complex systems.
- Patterns and principles for co-engineering Adaptive Complex Systems to achieve behaviors like Lean, Chargeback and Capacity alignment, Accountability, Competitiveness, and Innovation.
- Defining and deploying an IT solution using current and emerging technologies (sensor networks, mobile computing, service-oriented architectures) and related enterprise architecture patterns.
- Portfolio development and program management. Project specific presentations of research and best practices. Related guest lecturers from industry representing IT operations management and middleware technologies.
- Team project methodology and research presentations.

Topics in 794K include:

- Review of technologies available for enterprise integration.
- Project implementation plan and prioritization
- Development of non-functional requirements and request for proposals. Discussion of regulations (such as Sarbanes-Oxley, HIPA) and their impact on IT.
- Project implementations and review of work products. Application of architecture Tradeoff Analysis Method (ATAM), and Zachman Architecture. Case studies

covering the creation of an architecture representation and its use for analysis and identification of opportunities for enhancements.

- Project presentations.
- Guest lectures from industry representatives in IT operations management and middleware technologies.
- Innovation frameworks.
- Enterprise architecture patterns.

This course has an extensive project- and applied research-based foundation, thus making graduate students the primary audience of this course. All projects done in this course are through the CETI NSF-IUCRC program.

2.2 Evaluation of the courses

CSE 560: This course has an extensive set of learning outcomes as follows:

LO1. To master using and implementing each component of the the assemble-link-load-relocate-execute process.
LO2. To master using bit manipulation of integers and ascii characters to be able to emulate a simple computer that handles both integer and character I/O.
LO3. To master analyzing the intended audience for a written document and to write an audience profile.
LO4. To be familiar with group project organization techniques including conducting group meetings, recording minutes, and tracking project progress.
LO5. To be familiar with writing a relocating linking loader.
LO6. To be familiar with using different addressing modes.
LO7. To be familiar with subroutine linkage at the assembly level.
LO8. To be familiar with using compilers, debuggers, word processors, editors, diagram drawing programs, and profilers to design, build, and document a large software project.
LO9. To be familiar with using macros, including recursive and nested macros.
LO10. To be familiar with defining the purpose (persuade, inform, etc.) of a written document and select the appropriate rhetorical devices; and to write several pieces of documentation that have different purposes and to use appropriate organization to tie them together.
LO11. To be familiar with proof-reading own and others' writing.
LO12. To be familiar with emulating in software, the fetch-decode-execute cycle of a CPU.
LO13. To be familiar with the concept of a `machine' and its implementation via either translation or interpreation on lower level machines.
LO14. To be familiar with making engineering decisions involving tradeoffs (e.g., space-time tradeoffs in choosing a symbol table implementation).
LO15. To be familiar with the importance of communication skills, including oral, email, and other written documents such as meeting minutes.

LO16. To be familiar with software testing strategies including black-box versus white-box, unit testing, integration testing, top-down versus bottom-up testing, and construction and implementation of a test plan.
LO17. To be familiar with the economic and social forces that often drive technology to explain developments in system software.
LO18. To be familiar with using one structured approach to large software design to carry out a large group project.
LO19. To be exposed to issues in systems programming as opposed to applications programming.
LO20. To be exposed to memory management issues including caching, virtual memory, etc.
LO21. To be exposed to one-pass macro processing techniques.

CSE 560 meets multiple outcomes as the "keystone" course of the CSE curriculum. The technical content centers on 2 aspects: (a) tools that bridge the semantic gap between the application developer and the machine and (b) good software design and correct implementation of these tools using a high-level, object-oriented language. Thus, the learning outcomes above span both these aspects. A third aspect is technical writing – through the writing of two extensive sets of documentation – targeted at tool users and programmers. Thus outcomes LO10, LO11 and LO15 target this aspect.

CSE 601: The intended learning outcomes for this course are as follows:

LO1. Be familiar with social implications of decisions and actions of computing professionals.
LO2. Be familiar with the analysis of ethical issues facing computing professionals.
LO3. Be familiar with writing papers involving legal, ethical, and professional issues in computing.
LO4. Be familiar with making oral presentations, participating in formal debates, and in critically observing others' presentations and debates.
LO5. Be exposed to legal issues facing computing professionals.

The primary objective is for each participant to become informed and to develop reasoned opinions about the ethical, social, and legal dimensions of various situations that may be encountered by computer professionals, and about social issues that arise from computer technology. Participants will practice how to: hold an opinion, have reasons for their opinions, change opinions when compelled by best judgment, and rationally discuss opinions in cooperative pursuit of truth and good practices. Additional objectives are to develop communication skills, both written and oral, and to encourage ethical and responsible behavior.

CSE 757: The official learning outcomes for this course are as follows:

LO1. Be familiar with principles of modern software processes.
LO2. Be familiar with requirements analysis.
LO3. Master domain analysis using UML.

LO4. Master principles of object-oriented design using UML.
LO5. Master design principles and patterns.
LO6. Be familiar with principles of software testing for procedural and object-oriented software.

The learning outcomes of this course are in the process of being revised to better reflect its broad scope. An initial set of revised learning outcomes are below:

LO1. Be exposed to the trends impacting Enterprise Software Engineering
LO2. Be exposed to the need for frameworks for Enterprise Software Engineering
LO3. Be exposed to frameworks for analyzing the business context of enterprise IT systems, the concept of Business-IT alignment and related issues, and Enterprise Architecture frameworks for analyzing and achieving Business-IT alignment.
LO4. Be familiar with structured and agile software engineering frameworks; specifically structured and agile software engineering methodologies for requirements identification, analysis, architecture, design, deployment, testing, and project management.

A typical offering of CSE 757 requires students to conceive, architect and design <u>on paper</u> an enterprise scale software system in parallel with class lectures that present the relevant techniques with examples. Several simulations that illustrate the techniques are intended to be extensively used in this course. One such simulation, a Lego-based game that enables students to experience the use of Agile methodologies was designed and successfully conducted.

4 offerings of the course with the revised syllabus have taken place – 3 by Rajiv Ramnath and 1 by Bettina Bair.

CSE 758: This project-based Capstone course builds on CSE 757. The current learning outcomes are as follows:

LO1. Master software design, driven by non-functional requirements: scalability, security, usability and performance; generating and evaluating design alternatives; pattern-based software design.
LO2. Master the process of developing software: configuration management; project management, planning, team structure, roles and responsibilities; incremental and iterative, workbook-centered, object-oriented, agile, lean software development; project planning.
LO3. Master presenting work to an audience of peers.
LO4. Be familiar with business issues in software development.

The learning outcomes for this course are also in the process of being revised to better connect with its pre-requisite, CSE 757. An initial set of revised learning outcomes are below:

LO1. Be familiar with frameworks for analyzing the business context of enterprise IT systems, the concept of Business-IT alignment and related issues, and Enterprise Architecture frameworks for analyzing and achieving Business-IT alignment.
LO2. Master the principles underlying structured and agile software engineering frameworks, specifically structured and agile software engineering methodologies for requirements identification, analysis, architecture, design, deployment, testing, and project management.
LO3. Be familiar with the application of structured and agile software engineering frameworks, specifically structured and agile software engineering methodologies for requirements identification, analysis, architecture, design, deployment, testing, and project management.
LO4. Be familiar with the application of at least one industry-standard technology framework.
LO5. Master professional and formal presentations and communications to a varied set of stakeholders – customers, peers and superiors.

Students in CSE 758 achieve these outcomes by engaging in a quarter-long industry sponsored project and interacting with real customers.

CSE 502: These are the official learning outcomes and they reflect the intended audience for this course, i.e., (mainly graduate) students from other engineering and scientific disciplines (specifically non-CS majors) with some previous programming experience and no previous exposure to object-oriented languages, methodologies and technologies:

LO1. Master using control structures, built-in data types, and object-oriented program units of Java (interface, class, and package), and inter-unit relationships (implements and extends), to write programs for representative applications that are important in engineering and science.
LO2. Be familiar with using object variables (references/pointers), and with handling the problems they create compared to normal variables in procedural languages.
LO3. Be familiar with designing and developing new interfaces, classes, and packages of the kind that might arise in engineering and science applications.
LO4. Be familiar with using some of the most important object-oriented design patterns.
LO5. Be familiar with using UML class diagrams.
LO6. Be familiar with using formal specifications in interfaces.
LO7. Be exposed to the virtual machine model of modern computer systems.
LO8. Be exposed to software engineering issues.

In the class we put an emphasis on programming with interfaces (which has a natural connection with many things students want to do in Java) and the implications for strong typing, dynamic dispatching, and programming by contract. We also talk about inheritance, delegation, implements, visibility, and other object-oriented relationships involving classes, objects and interfaces. Basic design patterns are illustrated and applied to typical programming problems. UML is used as a design and discussion enabling graphical technique.

Eclipse was chosen as a development environment since it facilitates many of the software engineering techniques we also discuss, including refactoring, test driven development, design, group work, and documentation.

This course has been offered for close to a decade and has been taught in a relatively consistent fashion by various instructors over time. Perhaps the main change in recent years has been a heightened emphasis on development tools usually employed by professional Java programmers, e.g., Eclipse, Javadoc, JUnit, etc. These changes seemed appropriate given the audience and objectives for the course and were well received by the students.

CSE/ECE 668: Below are the official learning outcomes and they reflect the intended audience for this course, i.e., (mainly graduate) students from other engineering and scientific disciplines (specifically non-CS majors):

LO1. Be familiar with applying industry-standard software engineering technologies and tools to engineering and scientific software systems.
LO2. Be familiar with how to build a software system by "wiring together" software components
LO3. Be familiar with the efficiency issues connected with building software for engineering and scientific applications, and with how to use object orientation in the software without sacrificing efficiency.
LO4. Be familiar with the software design techniques needed to build such systems, such as using a standard modeling language, loosely coupled layered software architectures, event-driven programming, multithreading, design-for-reuse, and associated design patterns.

While this course was originally intended for non-CSE majors, there is strong interest from CSE majors.

CSE/ECE 767: The intended learning outcomes of this course are:

LO2. Be familiar with using UML-based industry-standard CASE tools.
LO3. Be familiar with following a process that is driven by the requirements of the users of the system.
LO4. Be familiar with model building, at the level of use cases, analysis, and design.
LO5. Be familiar with the important step of bridging the gap between analysis and design.
LO6. Be familiar with using objected-oriented design patterns.
LO7. Be familiar with the implications of building real-time software systems.

This course follows on from CSE 668 by taking the students through the requirements, analysis and design phases of the SDLC, thus providing the context for the implementation-oriented material in CSE 668. While this course was originally intended for non-CSE majors, there is strong interest from CSE majors.

CSE/ECE 769: The intended learning outcomes of this course are:

LO1. Master the technologies of enterprise computing that are most important in the software industry.
LO2. Master the advantages of architectures, specifically three tier architectures over two tier architectures.
LO3. Master how to build scalable distributed systems.
LO4. Master standards in describing data.
LO5. Be familiar with how to apply enterprise computing to scientific problems.
LO6. Be familiar with the importance of distributed computing through hands on experience.
LO7. Be familiar with the issues involved in enterprise mission critical applications.

This course is a strongly enterprise-technology oriented course. In the most recent offering of this course the technologies taught were XML and the Xerces XML toolkit, the J2EE suite of technologies – Servlets, Java Server Pages, Java Server Faces and Enterprise Java Beans. Also taught was the rich client technology, AJAX. Finally, 2 lectures covered the Enterprise Service Bus, using an open source ESB named ChainBuilder from a local company, Bostech (**http://www.bostechcorp.com**). As with CSE 668 and CSE 767, while this course was originally intended for non-CSE majors, there is strong interest from CSE majors. In fact, during the last offering of this course, there were 3 non-CSE majors out of the 45 students who enrolled in the class.

CSE 794J and 794K: The learning outcomes for this course sequence are:

LO1. Master techniques for representing and analyzing enterprise architectures in order to more effectively manage complex systems of business processes, organizations, and technologies.
LO2. Be familiar with the practice of applying architecture knowledge for developing strategic options for decision making and developing IT solution approaches for industry-sponsored problems.
LO3. Be familiar with communication and program management skills through making presentations that consider all stakeholder perspectives and techniques for identifying areas of IT innovation and requirements relevant to a particular organization.

This pilot 2-course sequence CSE 794J and K is on Enterprise Architecture, i.e. how complex organizations are structured and how technology supports the business. This course could fit in the Applied Software Engineering sequence, as an important course for computer science and other graduates seeking to enter professional practice in information technology, because of the relevance of organizational design and management, technology management, project and program management and vendor management to professionals. It is also a relevant course for CS graduates alone.

2.3 Relation to rest of the program

CSE 560: The prerequisites for CSE 560 are CSE 321 (or CSE 314 for *Information Systems* majors), CSE 360 (or ECE 265 for ECE majors), and a second writing course. The CSE 321 (or CSE 314) requirement stems from the large programming assignments that form the lab component of this course. Students come to CSE 560 with a strong foundation in the use and implementation of components built in C++ using the Resolve discipline. The ideas of tokenizers and parsers introduced in CSE 321 are extended in CSE 560. The CSE 360 (or ECE 265) requirement stems from the topic of the programming assignments: system software. Students are expected to be familiar with the basics of architecture and some assembly language.

CSE 560 serves as a direct prerequisite for the initial courses in several sub-disciplines: CSE 655 (programming languages), CSE 660 (operating systems), CSE 680 (algorithms), and CSE 681 (graphics). Of these, CSE 655 and CSE 660 require a conceptual model of the underlying hardware including memory and program execution. CSE 655 and CSE 681 both have extensive programming assignments, and assume the student is comfortable writing large programs.

CSE 560 also serves as a direct prerequisite for: CSE 731, CSE 772, and CSE 778.

Finally, CSE 560 serves as a prerequisite for two other courses in this course group: CSE 601 and CSE 757. Please refer to the description of these courses, below, for an evaluation of this prerequisite relationship.

CSE 601: The only prerequisite for this course is CSE 560. This course does not directly use any of the technical content of CSE 560. There are two reasons, however, for this prerequisite. Firstly, students in CSE 601 must complete a significant writing assignment (a term paper). They benefit, therefore, from the writing experience gained in CSE 560. Secondly, students in CSE 601 are expected to have a certain academic sophistication and general familiarity with many aspects of the discipline of computer science. Students should therefore be taking CSE 601 in their junior or senior year. The CSE 560 prerequisite is a simple mechanism to ensure that CSE 601 is not taken too early in the program.

CSE 601 is a prerequisite for all Capstone courses.

CSE 757: The only prerequisite for CSE 757 is CSE 560. There are two reasons that CSE 560 is a prerequisite for CSE 757. Firstly, CSE 757 builds on the experience that students have in CSE 560. CSE 560 serves as an introduction to many of the problems that occur during the software development and maintenance life cycle. It therefore motivates the techniques and solutions presented in CSE 757 for managing the complexities of this process. Secondly, students in CSE 757 are expected to have a certain academic sophistication and general familiarity with many aspects of the discipline of computer science.

CSE 757 also serves as the only prerequisite for CSE 758. See the next paragraph for a description of the relationship between these courses. CSE 757 serves as a prerequisite for CSE 772 (*Information Systems* majors may take CSE 616 to satisfy the CSE 772 prerequisite instead). Separate from the formal prerequisites, CSE 757 serves as a useful foundation for several of the other courses in the SE program--such as CSE/ECE 668, 767, and 769, as well as for the systems analysis course, CSE 616.

Given the large number of undergraduates who take CSE 757, we are considering changing its course number to CSE 657. This change will also better position CSE 757 as a valuable precursor to the Applied Software Engineering sequence.

CSE 758: The only prerequisite for CSE 758 is CSE 757. In CSE 757, students learn the principles and practice of software development. In CSE 758, students apply this knowledge in the design of a significant, large-scale system.

CSE 758 is not a prerequisite for any other course.

There has become a strong connection between CSE 758, and 2 other Capstones – CSE 762 (web-services Capstone) and CSE 772 (the database Capstone), through projects that are shared across all three courses.

CSE 502: The prerequisites for this course are calculus and introductory programming in any language and paradigm, which we would expect of most students in science and engineering; plus some additional programming experience beyond an intro course, which we would expect of students interested in studying applied software engineering if CS is not their major. CSE 502 is an alternate (i.e., non-CS major's) prerequisite for CSE/ECE 668. We also have made it an alternate prerequisite for some other courses as well in order to support the proposed ASE graduate minor. For example, non-CS majors may use it as a prerequisite for CSE 541 in place of CSE 221. This course is in many ways separate from the rest of the program since it "may not be used as a technical elective by CSE or CIS majors or minors." It is intended to give graduate students from other departments and advanced introduction to elementary topics that would enable them to take other advanced course in the CSE department. The formal prerequisites for CSE 502 are minimal--just introductory mathematics and a previous programming course. In actuality, the students are expected to have considerable experience with programming, just not programming taught in a formal way. Many gaps and misunderstandings have to be addressed.

CSE/ECE 668: This course is intended to be the next course in line following CSE 502. It is cross-listed with ECE 668, and is not open to students with credit for 694T/ECE694T or 768/ECE768.

CSE/ECE 767: In this course, students are taken through a case study using incremental and iterative use-case-driven process of building object-oriented scientific and engineering software systems; analysis, design, UML modeling, design patterns.

CSE/ECE 769: This is a course where the most current enterprise-scale technologies are taught. As such, the only requirement is knowledge of programming (i.e., CSE 560), but we require as pre-requisites CSC/ECE 668 and CSE 660 (the Introduction to Operating Systems course), the latter because there are concepts such as transactions and concurrency that would be helpful for the students to know beforehand.

CSE/ECE 794J and K: The prerequisites for the first course in this sequence are 682 or 731 or 758 or 762 or 772 or 778 or permission of instructor. Essentially, students must have attained a level of maturity in their understanding of computing applications.

In summary, the prerequisites for the courses in the group are satisfactory.

2.4 Evaluation with respect to CSE Program Outcomes

The courses in this group play a key role in meeting the *CSE* program outcomes. For these Outcomes we show each course's contribution at a learning outcome level, followed by a summary at a course level. For the EC 2000 outcomes we only show the contribution at the course level.

2.4.1 CSE Program Outcomes

As noted earlier, the BS-CSE POs are currently being revised. The (proposed) new POs are:

a. an ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, and engineering;
b. an ability to design and conduct experiments, as well as to analyze and interpret data;
c. an ability to design, implement, and evaluate a software or a software/hardware system, component, or process to meet desired needs within realistic constraints such as memory, runtime efficiency, as well as appropriate constraints related to economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability considerations;
d. an ability to function effectively on multi-disciplinary teams;
e. an ability to identify, formulate, and solve engineering problems;
f. an understanding of professional, ethical, legal, security and social issues and responsibilities;
g. an ability to communicate effectively with a range of audiences;
h. an ability to analyze the local and global impact of computing on individuals, organizations, and society;
i. a recognition of the need for, and an ability to engage in life-long learning and continuing professional development;
j. a knowledge of contemporary issues;
k. an ability to use the techniques, skills, and modern tools necessary for practice as a CSE professional.
l. an ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;

m. an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;
n. an ability to apply design and development principles in the construction of software systems of varying complexity.

2.4.2 Contribution to the CSE Outcomes at the Learning Outcome Level

In this section we detail the contribution of each course in the primary or “core” sequence to the CSE Learning Outcomes. Thus, CSE 502, 668, 767, 769 and 794J and K are not detailed here.

CSE 560: The relationship between the Learning Outcomes of CSE 502 and the CSE Outcomes are summarized in the table below:

Learning Outcome	**CSE (a)**	**CSE (b)**	**CSE (c)**	**CSE (d)**	**CSE (e)**	**CSE (f)**	**CSE (g)**	**CSE (h)**	**CSE (i)**	**CSE (j)**	**CSE (k)**	**CSE (l)**	**CSE (m)**	**CSE (n)**
LO1	XXX		XXX		XX						XX	XXX	XXX	XX
LO2	XXX		XXX		XX						XX	XXX	XXX	XX
LO3							XX							
LO4				XX										
LO5	XX										XXX	XX		
LO6	XX										XXX	XX		
LO7	XX										XXX	XX		
LO8	XX										XXX	XX		
LO9	XX										XXX	XX		
LO10							XXX							
LO11							XXX							
LO12	XX										XXX	XX		
LO13	XX										XXX	XX		
LO14	XX										XXX	XX		
LO15											XX	XX	XX	XX
LO16							XXX							
LO17			XX								XX	XX	XX	XX
LO18						X								
LO19			XX								XX	XX	XX	XX
LO20	XX										XXX	XX		

LO21	XX										XXX	XX		

CSE 601:

The relationship between the Learning Outcomes of CSE 601 and the CSE Outcomes are summarized in the table below:

Learning Outcome	**CSE (a)**	**CSE (b)**	**CSE (c)**	**CSE (d)**	**CSE (e)**	**CSE (f)**	**CSE (g)**	**CSE (h)**	**CSE (i)**	**CSE (j)**	**CSE (k)**	**CSE (l)**	**CSE (m)**	**CSE (n)**
LO1						XXX	X	XX		XX				
LO2						XXX	X	XX		XX				
LO3						XXX	X	XX		XX				
LO4						XXX	X	XX		XX				
LO5						XXX	X	XX		XX				

CSE 757:

The relationship between the revised learning outcomes of CSE 757 and the CSE Outcomes are summarized in the table below:

Learning Outcome	**CSE (a)**	**CSE (b)**	**CSE (c)**	**CSE (d)**	**CSE (e)**	**CSE (f)**	**CSE (g)**	**CSE (h)**	**CSE (i)**	**CSE (j)**	**CSE (k)**	**CSE (l)**	**CSE (m)**	**CSE (n)**
LO1			X						XX	XX				
LO2				XX		XX	XX		XX		XX			
LO3			XX	XX	XX	XX	XX	XX	XX		XX	XX		XX
LO4			XXX		XX	XX	XX	XX	XX		XX	XX		XX

CSE 758:

The relationship between the revised learning outcomes of CSE 758 and the CSE Outcomes are summarized in the table below:

Learning Outcome	**CSE (a)**	**CSE (b)**	**CSE (c)**	**CSE (d)**	**CSE (e)**	**CSE (f)**	**CSE (g)**	**CSE (h)**	**CSE (i)**	**CSE (j)**	**CSE (k)**	**CSE (l)**	**CSE (m)**	**CSE (n)**
LO1				XX	XX	XX	XX	XXX	XX	XXX	XX	XX		
LO2			X	XX	XX	XX	XX		X	X	XX	X	X	X
LO3			XXX	XX	XXX	XXX	XX	XX	XX	X	XXX	XXX	XXX	XXX
LO4			XXX		XXX				XXX		XXX	XXX	XXX	XXX
LO5							XXX		X	X				

2.4.3 Summary of Contribution to CSE Outcomes by the Courses

This group of courses as a whole strongly contributes toward meeting outcomes a, c, d, e, g, i, j, k, and l, contributes moderately to outcomes f, h and m, and contributes minimally to b.

Specifically, in CSE 560, students are exposed to software design and development for large systems, and they design their first large software system. CSE 560 thus contributes to program outcomes a, c, d, e, g, k, l, m and n. In CSE 757, they study the principles and practices that guide the development of software systems, and in CSE 758, they use these principles to design a software system. Further, in CSE 758, the projects come from real (external) clients and carry with them the full breadth of functional and nonfunctional requirements. Students tackle a wide variety of problems as they address a wide variety of considerations such as: correctness, UI usability, response time, throughput, scalability, and security. These 2 courses thus contribute to all outcomes *except* (b).

The "social, professional, and ethical considerations related to... computing" are exactly the substance of CSE 601. This course, therefore, contributes significantly to objective (f).

The software engineering sequence contributes significantly to objective (d) by repeatedly requiring students to work in teams. All assignments in CSE 560 are done by teams of 4 or 5 students. These assignments involve using a variety of skills to address the range of required tasks (implementation, quality control, documentation). Teamwork also plays a central role in CSE 758, although the precise structure of the teams varies between instructors. In some sections, the entire class has been structured as a matrix organization with cross-functional teams working towards a single unified goal. In other sections, the class has been strictly subdivided into independent teams each with similar or independent goals.

Written communication is a key component of many of the courses. CSE 560 focuses on technical writing, requiring extensive technical documentation of various aspects of the delivered product (e.g., use, maintenance, and testing). CSE 758 also has significant documentation requirements, and CSE 601 requires a term paper. Some sections of CSE 757 use written paper summaries where students report on a paper in the literature.

CSE 601, on the other hand, focuses primarily on oral communication, with weekly individual presentations and team debates. Students learn how to communicate their views effectively and persuasively in several ways. Firstly, some lecture time is set aside for instruction in the mechanics of presentation/rhetoric. Secondly, the instructor makes topic presentations similar in style to what is required of the students, so they have the opportunity to learn by example. Finally, students receive direct and immediate feedback both from the instructor and their fellow classmates. During debates, each audience member chooses which side to support and sits on the corresponding side of the room. As the debate progresses and audience members move back and forth, speakers see immediately which arguments are persuasive and which are not.

The software engineering sequence contributes significantly to objective (k). Students report that during their job interviews, they are asked to discuss many of the topics that they study in CSE 757, and that, because of what they have learned in the course, they are knowledgeable about the topics. Students also report that their potential employers are very impressed with the experience in software design that CSE 758 gives them.

The table below captures which courses specifically contribute to which program outcomes. In this table we use “XXX” to indicate a strong contribution, “XX” to indicate moderate contribution, and “X” to indicate minimal contribution. If a cell is empty, this indicates no contribution at all.

Course	**CSE (a)**	**CSE (b)**	**CSE (c)**	**CSE (d)**	**CSE (e)**	**CSE (f)**	**CSE (g)**	**CSE (h)**	**CSE (i)**	**CSE (j)**	**CSE (k)**	**CSE (l)**	**CSE (m)**	**CSE (n)**
560			XXX	X	XX		X		X		XX	XXX		XX
601						XXX	X	XX	X					
757	X		XX	X	XX	XX	X	XX	XX	X	XX	XX	X	XX
758	XX		XXX	XXX	XXX	XX	XXX	XX	XXX	XX	XXX	XXX	X	XXX

2.5 Feedback

The principal official vehicle for student feedback has been student evaluations in the form of SETs. These have generally been very positive, even for courses with high loads (CSE 560, CSE 757, CSE 758, CSE/ECE 769). These courses require a great deal of effort on the part of the students, but this effort seems to be worth the benefit reaped (and perceived by the students).

Unofficially, many students have communicated (either in person or by email) how useful the upper level software engineering courses have been in their subsequent employment. Several have observed that the CSE 757 background has been useful during interviews for securing second interviews and job offers. Students have also commented how invaluable the project experiences in CSE 758 have turned out to be in their employment.

A third form of feedback -- and one that is particularly important to the software engineering sequence -- is from practitioners in industry. In addition to alumni surveys that give general indications of the success of various aspects of the program as a whole, CSE 757 and CSE 758 have benefited in particular from a great deal of input and interaction with practicing software engineers.

Rajiv Ramnath was recognized in 2007 with the department's annual Outstanding Teaching Award. Paul Sivilotti had also received this teaching award in a prior year.

2.6 Major changes since the previous report

The last Software Engineering course group report was completed in June of 2005.

Addition of the Enterprise Architecture Sequence CSE 794J and CSE 794K: This course sequence was added to reflect the relevance of technology management, legacy software integration, organizational management, as well as deployment and infrastructure issues instead of a focus on just developing software.

Evolution in the Applied Software Engineering Sequence: CSE 769 was revised to stay abreast of current enterprise technologies. Specifically, XML frameworks, the complete J2EE framework, including EJB 3.0, Ajax, the Google Web Toolkit and the Enterprise Service Bus were incorporated.

Changes in CSE 757: Rajiv Ramnath, Jay Ramanathan and Neelam Soundarajan recently won an NSF CCLI award to examine how teachable curriculum may be mined from project experiences and the actual work-products from these industry projects. This work is now underway.

Changes in CSE 758: CSE 758 has continued to evolve, in particular as a result of the changes to CSE 757.

Changes in ECE/CSE 769: Since current technologies have evolved, what is taught in ECE/CSE 769 has changed, and will continue to do so.

2.7 Continuing and emerging concerns

Impact to CSE 560 following the emergence of CSE 421: With the establishment of CSE 421, there is the potential for CSE 560 to put a less emphasis on bringing the students up-to-speed on using a standard programming language and associated tools. Thus, CSE 560 is being considered for reduction to 4-credit hours, as well as for the incorporation of elements of software design (such as responsibility-driven design). However, since not all students taking CSE 560 have taken CSE 421, these changes are still being debated.

Establishing consistency across the CSE 757 sections: The opportunity provided by the NSF CCLI grant has enabled us to define the intellectual identity of CSE 757 and significantly revise and re-work it to match. This re-worked syllabus has been introduced in 3 offerings of CSE 757 (in Winter 2007, Fall 2008 and Winter 2009). As the experiences from teaching the new syllabus are fed back into the course material, CSE 757 will stabilize and the material will be disseminated to other instructors of CSE 757.

Given the rework of CSE 757, we believe that that a 600-level course number (such as 657) is a more accurate reflection of the content of a course that is taken primarily by undergraduates. This direction was suggested both by the Software Engineering course group report in Spring 1999 as well as the report in 2005.

Lack of a suitable textbook for CSE 757, especially in the light of the extensive re-work, remains an issue.

Finally, the inclusion of CSE 757 in the graduate “applied core” is being considered.

Move to semesters: The affect of the move to semesters will be seriously considered by the Software Engineering Course Group in the near future.

Changes in the IT industry:

The current economic downturn has the potential to affect both the hiring patterns of companies in the information technology and computing space, as well as the state of the professional practice in computing. This is an important phenomenon whose progress will be tracked, because it impacts what we teach in order to make CSE graduates competitive with respect to employability.

Conclusions

The Software Engineering course group is an important component of the *CSE* program. It plays an especially important role in meeting the program objectives and *EC 2000* Criterion 3 outcomes related to professional practice (eg teamwork and communication skills), professional ethics, and employability.

Course	**Coordinator**	**Recent Instructors**
CSE 502	Weide	Bucci, Mathis, Weide
CSE 560	Sivilotti	Giles, Heym, Ramnath, Sivilotti, Stutz
CSE 601	Mathis	Mathis
CSE 668	Crawfis	Crwafis, Khan
CSE/ECE 768	Khan	Khan
CSE 757	Ramnath	Bair, Bucci, Ramnath, Rountev
CSE 758	Ramnath	Ramanathan, Ramnath
CSE/ECE 767	Khan	Khan, Perry
CSE/ECE 769 (794R)	Sivilotti	Bucci, Ramnath, Sivilotti
CSE 794J and K	Ramanathan	Ramanathan

People involved in preparing report: Bettina Bair, Paolo Bucci, Roger Crawfis, Wayne Heym, Furrukh Khan, Bob Mathis, Doyt Perry, Jay Ramanathan, Rajiv Ramnath, Atanas Rountev, Paul Sivilotti, Bruce Weide

Rajiv Ramnath September 7, 2009