Department of Computer Science

STEPHEN L. BLOOM, DIRECTOR

FACULTY*

Professors

Stephen L. Bloom, Ph.D. (1968), Massachusetts Institute of Technology
Daniel Duchamp, Ph.D. (1988), Carnegie Mellon University
A. Satyanarayana, Ph.D. (1981), Jawaharlal Nehru Technological University

Associate Professors

Dominic Duggan, Ph.D. (1990), University of Maryland
George Kamberov, Ph.D. (1990), University of Pennsylvania
Aaron David Klappholz, Ph.D. (1974), University of Pennsylvania
David A. Naumann, Ph.D. (1992), University of Texas at Austin
John Oliensis, Ph.D. (1981) University of Chicago
Rebecca N. Wright, Ph.D. (1994) Yale University

Assistant Professors

Elli Angelopoulou, Ph.D. (1997), Johns Hopkins University
Adriana Compagnoni, Ph.D. (1995), Katholieke University Nijmegen
H. Quynh Dinh, Ph.D. (2002), Georgia Institute of Technology
Susanne Wetzel, Ph.D. (1998), Saarland University

**The list indicates the highest earned degree, year awarded and institution where earned.*

UNDERGRADUATE PROGRAMS

Computer Science

Computer science deals with the fundamental scientific laws and engineering principles which govern the design, manufacture and use of computers. A computer scientist is involved in work ranging from mathematical studies of problem-solving procedures to engineering research in advanced computing systems.

The undergraduate curriculum is designed to continue the Stevens tradition of a broad-based education exceeding the requirements of the Computer Science Accreditation Board (CSAB) in both hardware and software courses, along with a strong theoretical component as a foundation for software engineering. It features basic courses in chemistry, physics and mathematics; a sequence of courses in the humanities, as well as physical education; plus an in-depth sequence of specialized computer science courses.

The computer science courses at Stevens include the fundamentals needed by every computer scientist. As the software industry evolves, practitioners are increasingly expected to build reliable systems for mission- and life-critical applications. Such professionals distinguish themselves through mastery of scientific and mathematical foundations, mastery of software tools and methods, and experience in collaborative work on large projects. The Stevens computer science curriculum is designed to provide an outstanding education in all of these areas.

Upon completion of the program, you are able to formulate problems in algorithmic terms and solve them by means of a computer, as well as manage and reason about

large software systems, including the ability to use off-the-shelf components in the construction of software applications. You will have written multi-threaded programs, including thread synchronization, and will be able to reason informally about the correctness of programs. Electives provide you with an opportunity to delve deeper into specific areas of interest, including financial systems, intelligent design and manufacturing, mathematics and security.

Stevens' computer science graduates are greatly prized by industry and find excellent positions in areas ranging from the financial industry to software and telecommunication companies such as AT&T, IBM, Microsoft, Sun Microsystems and Verizon. In addition, a significant number of Stevens graduates continue their education at the graduate level, including those programs offered at Stevens.

The course sequence for computer science is as follows:

Freshman Year

Term I					Term II				
		Hrs. Per Wk. Class	Lab	Sem. Cred.			Hrs. Per Wk. Class	Lab	Sem. Cred
Ma 115	Math Analysis I	3	0	3	Ma 116	Math Analysis II	3	0	3
CS 115	Intro to Computer Science	2	2	3	CS 384	Data Structures & Alg. I	3	0	3
PEP 111	Mechanics	3	0	3	MA 334	Discrete Math	3	0	3
Ch 115	General Chemistry I	3	0	3	Ch 281	Biology & Biotechnology	3	0	3
Ch 117	General Chemistry Lab I	0	3	1	Ch 282	Biology Laboratory	0	3	1
Hu	Humanities	3	0	3	Hu	Humanities	3	0	3
PE 200	Physical Education I	0	2	1	PE 200	Physical Education II	0	2	1
	TOTAL	**14**	**7**	**17**		**TOTAL**	**15**	**5**	**17**

Sophomore Year

Term III					Term IV				
		Hrs. Per Wk. Class	Lab	Sem. Cred.			Hrs. Per Wk. Class	Lab	Sem. Cred
CS 383	Comp Org & Prog.	3	2	4	CS 488	Comp Architecture	3	0	3
CS 385	Data Str. & Alg. II	3	0	3	CS 434	Theory of Computation	3	0	3
CS 335	Computer Structures	3	0	3		Elective*	3	0	3
Hu	Humanities	3	0	3	Ma 222	Probability & Statistics	3	0	3
Hu	Humanities	3	0	3	Hu	Humanities	3	0	3
PE 200	Physical Education III	0	2	1	PE 200	Physical Education IV	0	2	1
	TOTAL	**15**	**4**	**17**		**TOTAL**	**15**	**2**	**16**

Junior Year

Term V					Term VI				
		Hrs. Per Wk. Class	Lab	Sem. Cred.			Hrs. Per Wk. Class	Lab	Sem. Cred
CS 492	Operating Systems	3	0	3	CS 442	Database Mgmt. Systems	3	0	3
CS 496	Programming Languages	3	0	3	CS 494	Compiler Design	3	0	3
	Elective*	3	0	3		Elective*	3	0	3
Hu	Humanities	3	0	3	Hu	Humanities	3	0	3
PE 200	Physical Education V	0	2	1	PE 200	Physical Education VI	0	2	1
	TOTAL	**12**	**2**	**13**		**TOTAL**	**12**	**2**	**13**

Senior Year

Term VII					Term VIII				
		Hrs. Per Wk. Class	Lab	Sem. Cred.			Hrs. Per Wk. Class	Lab	Sem. Cred
CS 551	Software Eng. & Pract. I	3	1	3	CS 552	Software Eng. & Pract. II	3	1	3
CS	CS Elective	3	0	3	CS	CS Elective	3	0	3
	Elective*	3	0	3		Elective*	3	0	3
	Elective*	3	0	3		Free Elective	3	0	3
Hu	Humanities	3	0	3	Hu	Humanities	3	0	3
	TOTAL	**15**	**1**	**15**		**TOTAL**	**15**	**1**	**15**

Humanities Electives must include at least one of these courses HPL 339, HPL 455, HSS 371, HHS 429.
* Must include a total of six electives, of which one must be Mgt 243, Mgt 111 or BT 121; one must be a science elective; and the remaining four must be CS courses or courses from a department approved Application Area sequence.

Suggested Application Areas

The Application Area sequences may devote up to four of the electives (marked by * in the requirements) to disciplines outside computer science. Depending upon the Application Area, the science elective and/or the management elective may be chosen to support the course sequence. In most cases, specific senior elective computer science courses are also chosen to complete the Application Area sequence.

You must receive prior departmental approval for each Application Area sequence. The computer science department works with other departments to develop Application Area sequences relevant to their disciplines. Below are Application Areas already approved.

Financial Systems

Intended for students who intend to make a career in the financial sector. Those who elect this Application Area option should take Mgt 243 Macroeconomics as their required management elective. Then the course sequence consists of the following:

Mgt 244 Microeconomics
BT 115 Financial Accounting
BT 215 Cost Accounting
BT 321 Finance

Mathematics

This Application Area focuses on topics in mathematics that utilize computing.
Ma 221 Differential Equations
Ma 331 Intermediate Statistics
Ma 346 Numerical Methods
Ma 520 Computational Linear Algebra I

Computational Chemistry & Biology

Ch 116/118 Chemistry II/Chemistry Lab II
Ch 321 Thermodynamics
Ch 381 Cell Biology
Ch 664 Computer Methods in Chemistry

It is also suggested, but not required, that Ch 484 Introduction to Molecular Genetics be taken as the free elective.

Engineering

For students with interest in hardware:

E 245	Circuits and Systems
CpE 358	Switching Theory and Logical Design
CpE 390	Microprocessor Systems
Ma 221	Differential Equations

In addition, students should take PEP 112 as the science elective.

Concentration Areas within Computer Science

Here are suggested course sequences for students who are interested in specific areas within computer science. These sequences are optional — indeed, each student may choose the six CS electives according to personal interests. A concentration does not appear on the diploma. Students should understand that concentrated electives are merely suggestions. A student may choose to take all, some, or none of the courses in a concentration.

Networks

CS 521 TCP/IP Networking
CS 549 Distributed Systems
CS 666 Information Networks I
CS 668 Foundations of Cryptography
CS 669 Network Management

Design of Games

A five-course concentration in Game Design is designed to prepare the student for an entry-level position in the computer-games industry. The emphasis is on the creation of network, multi-player and 3D games.

CS 437 Interactive Computer Graphics I
CS 482 Artificial Intelligence
CS 521 TCP/IP Networking
CS 638 Interactive Computer Graphics II
CS 549 Distributed Systems

It is further recommended that a game design be the subject of the student's CS 551/CS 552 software project.

Minor in Computer Science

You may qualify for a minor in computer science by taking the required courses indicated below. Completion of a minor indicates a proficiency beyond that provided by the Stevens curriculum in the basic material of the selected area.

Enrollment in a minor means you must meet the Institute's requirements for minors programs. Also, you may not use the same courses for both a major and a minor. Only courses completed with grade of "C" or better are accepted towards a student's minor.

Requirements for a Minor in Computer Science:

	Prerequisite
CS 115 Introduction to Computer Science	
Ma 334 Discrete Mathematics	
CS 383 Computer Org. & Programming	CS 384
CS 384 Data Structures & Algorithms I	CS 115, Coreq. Ma 334
CS 385 Data Structures & Algorithms II	CS 384

Interdisciplinary Program in Computational Science

For students interested in interdisciplinary science and engineering Stevens offers an undergraduate computational science program. Computational science is a new field in which techniques from mathematics and computer science are used to solve scientific and engineering problems. See the description of the Program in Computational Science in the Interdisciplinary Programs section.

Laboratories

Laboratories in the Department of Computer Science are used for course-related teaching and special problems, design projects and research. You are encouraged, and in many cases required, to gain hands-on experience with a number of different computers and computer operating systems and environments. The emphasis is to provide a broad understanding of the entire operation of the computer and its peripherals, and to teach you the use of modern tools. You are also exposed to a range of practical problems in laboratory assignments.

Research laboratories are also heavily involved in the undergraduate education for special course-related projects and for senior design. Laboratory facilities provide the latest equipment and computational facilities (hardware and software), and the department is continually involved in expanding, improving and developing new laboratories to provide you with the latest tools. For example, the Computer Science Laboratory has a collection of SGI, Sun and PC workstations.

GRADUATE PROGRAMS

Computer Science

Computer science is an intellectually challenging and exciting subject, having many applications to science, business and even the humanities. The graduate programs are designed to provide the student with a mastery of one broad area or a concentration in CyberSecurity or Quantitative Software Engineering (QSE). Alternatively, it is possible for a student to design a customized program, with the advice and approval of the department.

Master of Science – Computer Science

The departmental requirements for admission into the master's degree program in computer science are a bachelor's degree in computer science or computer engineering with a minimum grade point average (GPA) of 3.0 on a four-point scale. Applicants whose undergraduate degree is not in computer science or computer engineering must take the GRE General Test. In addition, such applicants must have completed at least one year of calculus and at least one programming course; they may be conditionally admitted subject to the completion of some or all of the Elements of Computer Science "ramp" courses (listed below) with a grade of B or better.

Elements of Computer Science

CS 550 Computer Organization and Programming*
CS 580 The Logic of Program Design*
CS 590 Introduction to Data Structures and Algorithms*
Ma 502 Mathematical Foundations of Computer Science*

**These courses may not be applied toward a graduate degree in Computer Science.*

Applicants from outside the U.S. or anyone seeking financial support must take the GRE General Test and the TOEFL. International students must demonstrate their proficiency in English by scoring at least 550 (or 210 for the computer-based test) on the TOEFL. Applicants desiring financial assistance are required to submit all application documents to Graduate Admissions before February 1 for fall admission, and before September 1 for spring admission.

All candidates for the Master of Science in Computer Science must complete the following core courses, plus six electives approved by the graduate academic advisor. Students choosing a concentration must complete the concentration courses listed below, plus elective courses (approved by their graduate academic advisor) to fulfill the requirements for the degree. Student must complete 30 credits (10 courses) of course work and earn a GPA of 3.0 or better.

Core Courses

CS 600 Analysis of Algorithms
CS 510 Theory of Programming Languages
CS 514 Computer Architecture
CS 520 Introduction to Operating Systems

Computer Science Concentrations

A student may choose one of the concentrations listed below or select electives from other areas of specialization with the approval of the academic advisor.

CyberSecurity Concentration

CS 573 Fundamentals of CyberSecurity
CS 668 Foundations of Cryptography
CS 693 Cryptographic Protocols
CS 694 E-Business Security and Information Assurance

And two electives, one of which must be CS 666 Information Networks I if the student has not taken it already.

Upon successful completion, the student will meet the requirements for a Master of Science in Computer Science (with a concentration in CyberSecurity indicated on the Transcript) and a Graduate Certificate in CyberSecurity.

Quantitative Software Engineering Concentration

CS 540 Fundamentals of Quantitative Software Engineering
CS 533 Cost Estimation and Metrics
CS 564 Software Requirements Acquisition and Analysis
CS 565 Software Architecture and Component-based Design
And, two electives with approval of the academic advisor.

Upon successful completion, the student will meet the requirements for a Master of Science in Computer Science (with a concentration in Quantitative Software Engineering indicated on the Transcript) and a Graduate Certificate in Quantitative Software Engineering.

Master of Science - Quantitative Software Engineering

The MS program in Quantitative Software Engineering emphasizes both practical software technology and the practical skills needed to apply software technology to the realization of high-quality software products on time and within budget. It is case-history and project-oriented, and teaches industry-standard practices and tools, including

the use of software process metrics. It is intended for those with more practical goals, as practicing software developers, as software managers or as software entrepreneurs. Students interested in developing new technologies or in technological or theoretical research should consider the MS in CS program.

Admission Requirements

Students must have an undergraduate degree with a GPA of 3.0 or better in Computer Science or Computer Engineering, or alternatively, an undergraduate degree in another field and on-the-job experience in software development. A solid working knowledge of a programming language is mandatory; C++ is the preferred language. Letters of recommendation should be specific as to the applicant's achievements rather than generally laudatory. Students with no software experience may be admitted subject to taking up to four of the Elements of Computer Science courses found earlier in the section (also referred to as 'ramp' courses.) They may also be required to take a course in C++ for credit.

Degree Requirements

Course requirements for the MS in QSE are:

CS 540 Fundamentals of Quantitative Software Engineering I
CS 533 Cost Estimation and Metrics
CS 564 Software Requirements Acquisition and Analysis
CS 565 Software Architecture and Component-Based Design
CS 567 Software Testing and Quality Assurance
CS 568 Software Project I
CS 569 Software Project II
Elective I
Elective II
Elective III

The three electives, chosen with the approval of the advisor, may be in software technology or in management. Recommended software technology topics include database management systems, distributed systems, system performance analysis, etc. Recommended management topics include project management, group dynamics, etc. Students should choose electives for which they have the stated prerequisites, some of which may not be among the required QSE courses. The 10 courses must be completed with a GPA of 3.0 or better.

Master of Science – Information Systems (Interdisciplinary) with Computer Science, E-Commerce, Information Security, Integrated Information Architecture, or Quantitative Software Engineering Concentrations

This program is designed to meet the increasing need for information technology professionals with both managerial and technical skills. It is an interdisciplinary program involving the School of Technology Management and the Computer Science department. For additional information about the Master of Science – Information Systems program, please refer to the School of Technology Management section of the catalog.

In addition to strong practical, real-world IT and management skills, graduates of the program leave with improved communication, interpersonal and team skills. The MSIS is a professional degree that integrates information and organizational cultures with emphasis on IT professionals who can contribute to the business.

Degree requirements include 12 graduate courses in each of the interdisciplinary MSIS tracks (36 credits) with a minimum GPA of 3.0. All require a bachelor's degree in Information Systems, Management or Computer Science. Work experience is considered when reviewing educational background. Students considering doctoral study are encouraged to complete a master's thesis as part of their degree.

Core Courses

Mgt 550 Introduction to Project Management
Mgt 623 Financial Management
Mgt 680 Organizational Behavior and Theory
Mgt 772 Analysis and Development of Information Systems
Mgt 780 Strategic Management of Information Technology
Mgt 781 Management of IT Organizations
Mgt 784 Integrating IS Technologies

Computer Science Track Concentration Courses

CS 561 Database Management Systems I
CS 540 Fundamentals of Quantitative Software Engineering I
CS 533 Cost Estimation and Metrics
CS 666 Information Networks I

Typical admission profile includes information systems technical career advancement, 3+ years information technology experience and a bachelor's in information systems or computer science. A strong mathematics and technical background is recommended.

E-Commerce Technical Track Concentration Courses

Core courses for this concentration are found in the School of Technology Management Section of the catalog.

Select 3 from:

CS 561 Database Management Systems I
CS 537 Interactive Computer Graphics I
CS 533 Cost Estimation and Metrics
CS 636 Integrated Services – Multimedia
CS 619 E-Commerce Technologies

Information Security Concentration Courses

Mgt 645 Cyber Security Principles
Mgt 646 Enterprise Architecture for Information Assurance
CS 573 Fundamentals of Computer Security
CS 694 E-Business Security & Information Assurance

Select *one* from:

Mgt 772 Analysis and Development of Information Systems
Mgt 773 Data Management
Mgt 776 Network Management

Integrated Information Architecture Track Concentration Courses

NIS 560 Introduction to Networked Information Systems
CS 561 Database Management Systems I
NIS 611 Digital Communications Engineering I
Mgt 773 Data Management

And, select *one* from the following:

CpE 654 Design and Analysis of Network Systems
CpE 592 Multimedia Network Security
CpE 636 Integrated Services - Multimedia
CpE 678 Information Networks I

Quantitative Software Engineering Track Concentration Courses

CS 540 Fundamentals of Quantitative Software Engineering I
CS 564 Software Requirements Acquisition and Analysis
CS 565 Software Architecture and Component-Based Design
CS 533 Cost Estimation and Metrics
Mgt 773 Data Management

Typical admission profile includes application systems analysis or testing career advancement, 3+ years information technology experience and a bachelor's in information systems or computer science. A strong mathematics and technical background is recommended.

Doctoral Program

Admission requirements to the Ph.D. program are naturally more stringent than those for the lesser degrees. To be admitted to the doctoral program, a student must have a master's degree in either computer science or computer engineering and a minimum GPA of 3.5 on a 4.0 scale. More attention is paid to the student's background and potential to perform independent research. All applications are considered individually. Exceptional students may be accepted after receiving the bachelor's degree. All Ph.D. applicants are required to take the GRE general and are strongly encouraged to take the CS subject examination. International students, except those from English-speaking countries, must submit minimum TOEFL scores of 550/210. Applicants are required to submit all application documents to Graduate Admissions before February 1 for fall admission and before September 1 for spring admission.

The Ph.D. degree requires 90 credits. A maximum of 30 credits can be applied toward the 90-credit requirement of the Ph.D. degree from a previous master's degree or from any other graduate courses, subject to the approval of the department. All Ph.D. candidates must take at least 30 credits of thesis work and at least 21 credits of course work at Stevens beyond the master's degree.

All Ph.D. candidates must pass the Ph.D. qualifying examination, which has two phases: a written examination and a project. Students may take the written examination only twice. Failure to pass the written examination in the second attempt will result in dismissal from the Ph.D. program. In addition, there is a required proposal defense and dissertation defense. For a detailed description of the requirements, request a copy of the doctoral requirements from the Department of Computer Science and the Graduate Student Handbook from the Office of Graduate Studies.

After the student has successfully completed the qualifying and preliminary examinations, s/he must choose an advisor. The advisor must be a full-time Computer Science faculty member. However, an outside co-advisor may be chosen subject to the approval of the Ph.D. program committee and the department director.

Once a suitable topic has been found and agreed upon with the advisor, the student must prepare a thesis proposal. The proposal must indicate the direction the thesis will take and procedures that will be used to initiate the research. Ordinarily some preliminary

results are included in the proposal. In addition, the proposal must indicate that the student is familiar with the research literature in his/her area. To this end, the proposal must include the results of a thorough literature search. A committee of at least three faculty members must accept the written dissertation proposal. The committee chairperson is the dissertation advisor. The other two members should be Computer Science department faculty. After the written proposal has been accepted, the examination committee conducts an oral defense. At this defense, the student presents his/her proposal.

All Ph.D. candidates who are working on a dissertation must have a doctoral dissertation advisory committee, chaired by the dissertation advisor, consisting of at least four members. The dissertation advisor and at least two other members must be full-time faculty of the Computer Science department. The committee may include members from other institutions. In addition, a regular faculty member with another department at Stevens must also be a member of the committee. The committee must approve the completed dissertation unanimously. After the dissertation has been approved, it must be defended in a seminar open to the public.

Graduate Certificate Programs

The Computer Science department offers graduate certificate programs to students meeting the regular admission requirements for the master's program. Each Graduate Certificate program is self-contained and highly focused, carrying 12 or more graduate credits. Most of the courses may be used toward the master's degree as well as for the certificate. Those who enroll in the Database Systems program are expected to have a working knowledge of C++. Students who do not have such knowledge will be admitted, but are required to take CS 590 as a prerequisite.

Computer Graphics

CS 600 Analysis of Algorithms
CS 537 Interactive Computer Graphics I
CS 638 Interactive Computer Graphics II
CS 558 Computer Vision

CyberSecurity

CS 573 Fundamentals of CyberSecurity
CS 668 Foundations of Cryptography
CS 693 Cryptographic Protocols
CS 694 E-Business Security and Information Assurance

Database Systems

CS 674 Theory of Object-Oriented Software Design
CS 561 Database Management Systems I
CS 562 Database Management Systems II
CS 563 Object-Oriented Database Management Systems

Distributed Systems

CS 520 Introduction to Operating Systems
CS 521 TCP/IP Networking
CS 549 Distributed Systems
CS 666 Information Networks I

Quantitative Software Engineering

Students may choose any **four** of the following courses:

CS 540 Fundamentals of Quantitative Software Engineering
CS 533 Cost Estimation and Metrics
CS 564 Software Requirements Acquisition and Analysis
CS 565 Software Architecture and Component-based Design
CS 567 Software Testing, Quality Assurance and Maintenance
CS 551 Software Engineering and Practice I
CS 552 Software Engineering and Practice II

Software Design

CS 565 Software Architecture and Component-based Design
CS 567 Software Testing, Quality Assurance and Maintenance
CS 600 Analysis of Algorithms
CS 765A Selected Topics in Computer Science– C++

Theoretical Computer Science

CS 600 Analysis of Algorithms
CS 601 Algorithmic Complexity
CS 630 Automata and Formal Languages
CS 634 Decidability and Computability

Elements of Computer Science

CS 550 Computer Organization and Programming*
CS 580 The Logic of Program Design*
CS 590 Introduction to Data Structures and Algorithms*
Ma 502 Mathematical Foundations of Computer Science*

**These courses may not be applied toward a graduate degree in computer science.*

LABORATORIES

Laboratories in the Department of Computer Science are used for course-related teaching and special problems, design projects and research. Students are encouraged, and in many cases required, to gain hands-on experience with a number of different computers and computer-operating systems and environments. The emphasis is to provide a broad understanding of the entire operation of the computer and its peripherals, and to instruct students in the use of modern tools. Students are exposed to a range of practical problems in laboratory assignments.

Research laboratories are also heavily involved in both undergraduate and graduate education with special project and dissertation projects. All research laboratories serve a dual-use function. Undergraduate students use these facilities through special course-related projects and for senior design. Graduate students pursue course-related projects and thesis research.

The laboratory facilities provide the latest equipment and computational facilities (hardware and software). The department is continually involved in expanding, improving and developing new laboratories to provide the students with the latest tools. Both undergraduate and graduate curriculums are continually reviewed and updated to insure that we provide the current methodologies and technologies and to maintain the standards that are incorporated in Stevens' goals.

Computer Science Laboratory

The Computer Science Lab has a collection of SGI, Sun and PC workstations, including a four-processor Sun server and a four-processor SGI Origin server. These UNIX workstations are available to the entire Stevens community for education, research and even recreation. In addition, many courses make use of the lab for programming purposes.

Computer Vision Laboratory - Dr. Elli Angelopoulou

The primary objective of the research performed in the Computer Vision Laboratory is to apply rigorous physical and mathematical principles towards image interpretation. The work performed in the lab is multidisciplinary, combining many diverse academic disciplines, including physics, mathematics, engineering and, above all, computer science. Some of the major thrusts in the lab include photometry, 3D-shape reconstruction, shape analysis, object recognition and multispectral imaging.

The Computer Vision Laboratory offers students a hands-on experience with image capturing and processing equipment. There is a dedicated workstation that is mainly used for the capture of still images and movies. The environmental conditions in the lab are strictly controlled. If needed, the lab can become a dark room. An optic table allows for the precise positioning of equipment. A collection of optical components allows for experimentation with enhanced image capture. The lab has its own server and UNIX workstations for storing, processing and analyzing images.

One of the industrial sponsors of the lab is Siemens Corporate Research, whose Imaging and Visualization Department has already donated a wide collection of optical equipment. There is also ongoing collaboration with the General Robotics Automation and Sensory Perception (GRASP) Laboratory at the University of Pennsylvania.

Laboratory for Secure Systems - Dr. Adriana Compagnoni, Dr. Dominic Duggan, Dr. David Naumann, Dr. Susanne Wetzel, Dr. Rebecca Wright

The Lab's mission is to pioneer new technologies for high-assurance and secure systems and prototype tools that can provide guarantees that a system will not exhibit unpredictable behavior in a hostile environment. Our objective is to consolidate and organize research and tool-building efforts already under way at Stevens. The Lab is funded by grants from the New Jersey Commission on Science and Technology, the National Science Foundation and the Stevens Institute of Technology Technogenesis Fund. The facilities of the Lab include several desktop machines, PDAs with wireless Ethernet, and Bluetooth devices for experimentation. The Lab is affiliated with the New Jersey Institute for Trustworthy Enterprise Software.

Part of the research work is focused on building better trust models for components. Some of this work is using static analysis techniques to check access control and information flow properties for untrusted components. There is also work on pushing type safety from high-level languages down to the assembly language level, and in the process checking properties of heap space usage. Other work has been performed on type systems for dynamic linking and "hot" updates of program libraries at run-time.

Another thrust of the work in the Lab has been in network security, particularly for wireless networks. Work continues on attacks that can be mounted on ad-hoc wireless networks, and in the design of new authentication and key establishment protocols that can be used to improve the security of wireless communication in general. Recent work has also looked at type-based approaches to cryptography, to specify and ensure trustworthiness guarantees for communication channels.

Software Engineering Laboratory - Mr. Larry Bernstein, Dr. David Klappholz

The Laboratory for Quantitative Software Engineering, supported by grant money from the New Jersey Commission on Science and Technology, has six workstations of which five are dual bootable to either Windows 2000 or SuSe Linux. All six workstations have AMD Athlon™ processors with 256mb RAM. The workstations are connectable as both Ethernet and wireless LANs.

The lab's use is two-fold; first, it is used by those students in the required two-semester Senior Design sequence whose projects are more profitably implemented on networked workstations than on personal laptops, with which all Stevens students are equipped. A special feature of the Senior Design course is that it is taught using a novel pedagogic methodology entitled Live-Thru Case Histories. Further development of, and the study of the efficacy of, the Live-Through Case History Method, are being studied under grants from the New Jersey Commission on Science and Technology and the National Science Foundation, in part with the aid of custom software being developed in the lab.

Second, as affiliates of both Prof. Barry Boehm's University of Southern California Center for Software Engineering and of the DOD-sponsored CEBASE (Center for Experimentally-based Software Engineering) led by Prof. Boehm and Prof. Victor Basili of the University of Maryland, the Software Engineering Lab's faculty are using the lab for experimentation in software engineering technologies and methodologies. Subjects of these studies include high-reliability software, methods for avoiding the need to perform full-scale defect detection and elimination, and modern agile software development practices pair programming, refactoring, etc.

Visualization Laboratory (VLab) - Dr. George Kamberov

The research in the VLab is in the general areas of visualization, computer graphics and computer vision with applications in medical imaging and diagnostics, cell biology, scientific computing, robotics and computational finance. Current research projects include the development of new geometric methods and efficient computational algorithms for representation, recognition and visualization of surface shapes and shape deformations, and for pre-compression data reduction in visual data communications.

The VLab is part of the mV2 (multimedia vision and visualization) group, and has close ties with the Vision Lab at Stevens.

New Jersey Center for Software Engineering

Members of the Computer Science Department hold a large grant from the New Jersey Commission on Science and Technology (NJCS&T), focused on research in software engineering aspects of networks and distributed programming. This grant is held jointly with New Jersey Institute of Technology (NJIT) and Rutgers University, New Brunswick.

Based at Stevens, the New Jersey Center for Software Engineering (NJCSE) was founded in mid-2000 as the corporate outreach (technology transfer) arm of this research activity. Academic institutions affiliated with NJCSE are Stevens Institute of Technology, New Jersey Institute of Technology, Rutgers University in New Brunswick and Monmouth University.

NJCSE activities for 2001 included regular technical meetings with Stevens, Rutgers and NJIT researchers and industry representatives. NJCSE activities also included a Student Project Showcase and a repeat of the Career Opportunities Program. As of January 1, 2001, industry affiliates included Avaya, Telcordia, Rational and IBM. NJCSE offers companies state-of-the-art technical programs and early access to some of the best graduating computer science students in New Jersey.

UNDERGRADUATE COURSES

CS 115 Introduction to Computer Science
(2-2-3)
An introduction to the basic concepts of computer systems, information networks and programming. To cultivate the core competencies required for effective use of applications as well as for programming, the course focuses on algorithmic thinking and problem solving. An intensive lab augments the classroom exercises with hands-on programming projects. Topics covered are: computer networks; Internet Protocol; WWW e-mail; multimedia; applets and mobile code; using the web to provide and access information; basic computer architecture; CPU; ALU; memory cache; primary and secondary storage; data representation; fetch-decode-execute cycle; Boolean logic; combinational circuits; algorithmic thinking and problem-solving; analyzing requirements; algorithm design; functions and procedural abstraction; pre- and post-conditions; loops and invariants; top-down design; libraries and simple data structures (arrays and lists); tools and techniques for debugging and testing; software architecture; classes and objects; and encapsulation.

CS 181 Introduction to Computer Science Honors I
(3-2-4)
In-depth introduction to programming in Java and to Computer Science in general. The course emphasizes structured programming techniques. Included is a discussion of computer systems from the hardware and systems software point of view. The Java programming concepts covered include: standard data types; structure of a Java program; input and output; graphical user interfaces; arithmetic and Boolean operations; assignment operations; standard functions; branching statements (if/else and switch statements); iteration operations (while, do/while and for statements); functions; value and reference parameters, local variables; scope of variables and constants, function prototypes and overloading; arrays; pointers; and structures. By invitation only. Students who complete this class are exempt from CS 384.

CS 182 Introduction to Computer Science Honors II
(3-0-4)
Advanced programming concepts covering classical data structures and object-oriented programming. Emphasis is on building a collection of re-usable software components that will form the basis of future programming efforts. The data structures covered include lists, stacks, queues, trees, binary search trees and balanced search trees. The object-oriented features of Java covered include classes, templates, inheritance, polymorphism and run-time binding. Also included is a discussion of the analysis of asymptotic running times of algorithms. Computer Science and Computer Engineering students who complete CS 182 are exempt from taking CS 385. Prerequisite: CS 181 and permission of the instructor.

CS 335 Computational Structures
(3-0-3)
This course uses functional programming to study discrete mathematics, building on Ma 334. It begins with an introduction to a small subset of a functional language used in the course to explore examples and implement some basic algorithms. Operations on relations are explored by the use of lists to represent finite sets, functions and relations, as well as operations of composition, transitive closure, direct image, etc. Some algebraic structures, such as monoids and semirings, are introduced; lists and relations are used as primary examples of monoids; and trees

are introduced as inductive data types. Other topics: structural recursion and induction; abstract syntax as data type; interpreters for simple languages (monoid expressions; Boolean expressions; integer expressions; relational expressions) without variable binding; inductively defined relations; transition rules for configurations of simple machine; Partial orders and lattices; simple abstract interpretations of expressions; equational logic. Prerequisite: Ma 334.

CS 381 Switching Theory and Logical Design
(3-0-3)
Digital systems; number systems and codes; Boolean algebra; application of Boolean algebra to switching circuits; minimization Boolean functions using algebraic, Karnaugh map and tabular methods; design of combinational circuits; programmable logic devices; sequential circuit components; design and analysis of synchronous and asynchronous sequential circuits. Cross-listed with CpE 358. Prerequisite: CS 115 or CS 181.

CS 383 Computer Organization and Programming
(3-2-4)
Basic structure of the stored program computer, addressing methods and program sequencing, instruction sets and their implementation, the CPU and microprogrammed control, input/output organization, peripherals and interfacing, and main memory. Detailed study of a small machine. The laboratory is devoted to assembly language programming. Prerequisite: CS 182 or CS 384.

CS 384 Data Structures and Algorithms I
(3-0-3)
Introduction to basic data structures and algorithms. Emphasis is placed on programming in C++ and debugging skills. Topics include: control flow, loops, recursion; elementary data structures (lists, stacks, queues) and their implementation via arrays and pointers; primitive sorting algorithms; binary trees and searching. Prerequisite: CS 115 or CS 181. Corequisite: Ma 334.

CS 385 Data Structures and Algorithms II
(3-0-3)
A continuation of CS 384, focusing on algorithm development, including running time analysis and correctness arguments. Topics include: asymptotic notation and running time analysis; program verification using loop invariants; advanced sorting algorithms, linear sorting algorithms, lower bounds; general trees, priority queues and heaps; set implementations; elementary graph algorithms. Prerequisite: CS 384 or CS 182.

CS 434 Theory of Computation
(3-0-3)
Introduction to the mathematical theory of computation. A number of models of computation are considered, as well as their relation to various problem classes (e.g. solvable problems, polynomial time solvable problems). Topics relevant to computer science such as the theory of formal languages and low complexity classes are particularly emphasized. Prerequisite: CS 335 and Ma 334.

CS 437 Interactive Computer Graphics
(3-0-3)
A comprehensive introduction to the field of Computer Graphics. Students study the conceptual framework for interactive computer graphics: transformations, viewing, shading, clipping, rasterization, curves and surfaces, selected topics.

OpenGL is used as an application-programming interface. Prerequisite: CS 385.

CS 442 Database Management Systems (3-0-3)
Introduction to the design of relational databases and the use of standard relational query languages. Topics include: relational schemas; keys and foreign key references; relational algebra (as an introduction to SQL); SQL; Entity-Relationship (ER) database design; translating from ER models to relational schemas and from relational schemas to ER models; functional dependencies; normalization. Prerequisite: CS 385.

CS 482 Artificial Intelligence (3-0-3)
Overview and history; LISP and Prolog; applications; search, game-playing, logic and theorem proving, expert systems, natural language processing, learning, robotics, computer vision. Prerequisite: CS 385.

CS 487 Digital System Design (3-0-3)
Digital design concepts, building blocks for digital systems and digital system organization. Design of combinational logic circuits and arithmetic functions. Approaches to sequential circuit design, separation of data and control, the algorithmic state machine. System controller design: System controllers using MSI/LSI circuits. Clocks and timing. Asynchronous systems design. Metastability and reliability. Cross-listed with CpE 487. Prerequisite: CS 381.

CS 488 Computer Architecture (3-0-3)
An introduction to the functional level structure of modern pipelined processors and the empirical and analytic evaluation of their performance. Topics include: empirical and analytic techniques for measuring performance (use of various means, Amdahl's Law, benchmarks); tradeoff analysis; principles of instruction set design and evaluation (memory addressing, operations, types and sizes of operands, instruction set encoding, CISC vs. RISC and related compilation issues); pipelining (basics, data hazards, control hazards); memory systems. Prerequisite: CS 383. Corequisite: Ma 222.

CS 492 Operating Systems (3-0-3)
The use and internals of modern operating systems. Lectures focus on internals whereas programming assignments focus on use of the operating system interface. Major topics include: the process concept; concurrency and how to program with threads; memory management techniques including virtual memory and shared libraries; file system data structures; and I/O. Prerequisites: CS 383 and CS 385.

CS 494 Compiler Design (3-0-3)
Design and implementation of compilers, principles of languages translation. Each student implements a complete compiler for a small but substantial language. The stages of a compiler. Boot-strapping a compiler. Lexical analysis, regular expressions, finite state machines. Syntactic analysis, context free grammars, parsers. Semantic analysis, type checking, symbol tables. Syntax-directed translation. Data flow analysis, peephole optimization. Code generation. Prerequisite: CS 385.

CS 496 Programming Languages (3-0-3)
An introduction to programming language design and implementation, with an emphasis on the abstractions provided by programming languages. Assignments

include implementing one or more interpreters for simple procedural languages, in a functional language such as Scheme or ML. Recursive types and recursive functions; structural induction. Abstract data types. Abstract syntax. Implementing languages with interpreters. Static vs. dynamic scoping, closures, state. Exceptions. Types: type-checking, type inference, static vs. dynamic typing. Object-oriented languages: classes and interfaces, inheritance and subtyping. Polymorphism and genericity. Prerequisites: CS 335 and CS 385.

CS 498 Senior Research I
(0-8-3)
Individual research project under the guidance of a faculty member of the department, whose prior approval is required. Either a written report in acceptable journal format or the completion of a senior thesis, as well as an oral presentation is required at the end of the project. Senior students only. CS 498 and CS 499 cannot be taken simultaneously.

CS 499 Senior Research II
(0-8-3)
Individual research project under the guidance of a faculty member of the department, whose prior approval is required. Either a written report in acceptable journal format or the completion of a senior thesis, as well as an oral presentation is required at

GRADUATE COURSES

All Graduate courses are 3 credits except where noted.

Computer Science
Undergraduate students may take any 500-level course for which they satisfy the prerequisites. Higher-numbered graduate courses may be taken only by undergraduates who satisfy the prerequisites, have a 3.0 GPA and receive permission of the instructor. Note that prerequisites for graduate courses are stated in terms of other graduate courses, and that equivalences exist between certain undergraduate and graduate courses:

MA 502 = MA 334 Discrete Mathematics
CS 590 = CS 385 Data Structures & Algorithms II
CS 537 = CS 437 Interactive Computer Graphics
CS 561 = CS 442 Database Management Systems
CS 514 = CS 488 Computer Architecture
CS 520 = CS 492 Operating Systems
CS 516 = CS 494 Compiler Design
CS 510 = CS 496 Programming Languages
CS 550 = CS 383 Computer Organization and Programming
CS 570 = CS 115 Introduction to Computer Science
CS 580 = CS 384 Data Structures and Algorithms I

In fulfilling their study plans, undergraduates should always take the undergraduate course whenever a choice exists.

the end of the project. Senior students only. CS 498 and CS 499 cannot be taken simultaneously.

CS 501 Introduction to JAVA Programming
An introduction to the Java programming language for those students who have little or no programming background. It is intended as an elective for the Master of Science in Information Systems to be taken near the end of the program. Basic topics considered will be programs and program structure in general and Java syntax, data types, flow of control, classes, methods, and objects, arrays, exception handling, and recursion. In addition, the use of Java in enterprise-wide computing and distributed systems will be introduced by considering APIs in general, and the ones specific to JDBC and the Java security features in particular. *Not for credit for Computer Science department majors.*

Ma/CS 503 Discrete Mathematics for Cryptography
Topics include basic discrete probability including urn models and random mappings; a brief introduction to information theory; elements of number theory including the prime number theorem, the Euler phi function, the Euclidean algorithm, the Chinese remainder theorem; elements of abstract algebra and finite fields including basic fundamentals of groups, rings, polynomial rings, vector spaces and finite fields. Carries credit toward the Applied Mathematics degree only when followed by CS 668. Recommended for high-level undergraduate students. Prerequisite: Ma 502.

CS 505 Probability and Stochastic Processes I
Axioms of probability. Discrete and continuous random vectors. Functions of random variables. Expectations, moments, characteristic functions and moment generating

functions. Inequalities, convergence concepts and limit theorems. Central limit theorem. Characterization of simple stochastic processes: wide-sense stationality and ergodicity.

CS 510 Theory of Programming Languages
Principles of functional, imperative and object-oriented programming languages; elements of language theory; the typed-l calculus, functional languages, stack implementation of recursion; imperative languages, block structure, more on stack allocation model; user-defined types, heap storage model; object-oriented languages, data abstraction, genericity, polymorphism, inheritance. Case studies may include Algol, Pascal, Ada, LISP, Scheme, Smalltalk, Java, C++. Prerequisite: Undergraduates: CS 335 and CS 385; Graduates: Ma 502 and CS 590.

CS 511 Concurrent Programming
The study of concurrency as it appears at all levels and in different types of computing systems. Topics include: models of concurrency; languages for expressing concurrency; formal systems for reasoning about concurrency; the challenges of concurrent programming; race conditions; deadlock; livelock and nondeterministic behavior; prototypical synchronization problems such as readers-writers and dining philosophers; mechanisms for solution of these problems, such as semaphores, monitors, and conditional critical regions; important libraries for concurrent programming; message passing, both synchronous and asychronous; applications of multi-threaded concurrent programming, and parallel algorithms. Substantial programming required. Prerequisites: CS 492, CS 520, or equivalent.

CS 514 Computer Architecture
Measures of cost, performance, and speedup; instruction set design; processor design; hard-wired and microprogrammed control; memory hierarchies; pipelining; input/output systems; additional topics as time permits. The emphasis in this course is on quantitative analysis of design alternatives. Prerequisite: CS 550 or equivalent. Cross-listed with CpE 514 and NIS 514.

CS 516 Compiler Design
Lexical analysis; syntax analysis; symbol table construction; semantic analysis; syntax-directed translation; dataflow analysis; liveness analysis; register allocation. The emphasis in this course is on the integration of the various parts of a compiler. Each student writes a complete compiler for a small but substantial language. Prerequisite: CS 510 or equivalent.

CS 520 Introduction to Operating Systems
Overview of operating systems. CPU scheduling; process concept, suspend and resume, interrupt processing, multiprogramming. Concurrent processes; the critical section problem, mutual exclusion, semaphores, process coordination, communication and synchronization. CPU scheduling algorithms. Deadlocks; conditions for deadlock, deadlock detection and avoidance. Memory management; memory management schemes—single contiguous, static and dynamic partitions, relocatable partitions, paging, demand paging, page replacement strategies. Secondary storage management; disk scheduling concepts and algorithms. Students will be given programming assignments on a regular basis. This course is not centered on any particular operating system or hardware; instead, it covers fundamental concepts that are applicable to a variety of systems. Prerequisites: CS 590 and CS 550 or their equivalent.

CS 521 TCP/IP Networking
Introduction to IP networking.

Examination of all layers of the OSI stack. Detailed examination of the IP, ICMP, UDP and TCP protocols. Basic concepts of network design: end-to-end principle, routing, encapsulation, flow control, congestion control and security. Detailed coverage of TCP. Some treatment of important Internet applications and services. Emphasis on network layer and above. Assignments focus on protocols and software. Prerequisites: CS 492 or CS 520.

CS 527 Logical Design of Digital Systems I
Design concepts for combinational and sequential (synchronous and asynchronous) logic systems; the design processes are described algorithmically and are applied to complex function design at the gate and register level; the designs are also implemented using software development tools—logic compilers for programmable logic devices and gate arrays. Cross-listed with CpE 643.

CS 528 Logical Design of Digital Systems II
The design of complex digital logic systems using processor architectures. The architectures are implemented for reduced instruction set computers (RISC) and extended to complex instruction set computers (CISC). The emphasis in the course is the design of high-speed digital systems and includes processors, sequencer/controllers, memory systems and input/output. Prerequisites: CS 514, CS 527.

CS 533 Cost Estimation and Metrics
The course deals with the management of software projects through the use of objective metrics which help developers and managers to understand the scope of the work to be accomplished, the risks which will occur, the tasks to be performed, the resources and effort to be expended, and the schedule to be observed. It provides the student with a thorough introduction to facility with, and understanding of, such industry-standard software sizing metrics as Function, Feature and Object Points and their relationship to the lines-of-code metric. It provides the student with a thorough introduction to and understanding of such industry-standard software estimation tools such as COCOMO II and KnowledgePlan. Prerequisites: CS 540 or CS 551.

CS 535 Financial Computing
This is a course in modeling the values of assets and financial derivatives and the software implementation of these models for pricing, simulations and scenario analysis. The course includes an introduction to markets and financial derivatives, and a development of the necessary tools from the theories of stochastic processes and parabolic differential equations. An integral part of the course is the use of financial information sources and software packages available on the Internet for modeling and analysis. Prerequisite: Acquaintance with Multivariable calculus and programming in C++ and/or Java.

CS 537 Interactive Computer Graphics I
This is an introductory-level course to computer graphics. No previous knowledge on the subject is assumed. The objective of the course is to provide a comprehensive introduction to the field of computer graphics, focusing on the underlying theory and thus providing strong foundations for both designers and users of graphical systems. The course will study the conceptual framework for interactive computer graphics; introduce the use of OpenGL as an application programming interface (API) and cover algorithmic and computer architecture issues. Cross-listed with CpE 537.

Prerequisite: Graduates: CS 590; Undergraduates: CS 385.

CS 540 Fundamentals of Quantitative Software Engineering I
This course introduces the subject of software engineering, also known as software development process or software development best practice from a quantitative, i.e., analytic- and metrics-based point of view. Topics include introductions to: software life-cycle process models from the heaviest weight, used on very large projects, to the lightest weight, e.g., extreme programming; industry-standard software engineering tools; teamwork; project planning and management; object-oriented analysis and design. The course is case-history and project oriented. Prerequisites: Admission to the MS in CS program and completion of any required ramp courses, OR admission to the MS in QSE program. Undergraduates may take this course if they have senior status, or have obtained the written permission of the instructor. CS 540 and CS 551 may not both be taken. Cross-listed with CpE 540.

CS 542 Fundamentals of Quantitative Software Engineering II
This course is a project-oriented continuation of CS 540. It is intended for computer science majors interested in learning the software development process, but not interested in the full MS program in QSE or the Graduate Certificate in QSE. Students who have taken the defunct CS 642 Software Engineering II, CS 568 and/or CS 569 may not take this course for credit. Prerequisite: CS 540. Cross-listed with CpE 542.

CS 549 Distributed Systems
Fundamental characterization of computer networks and distributed systems; network programming using sockets and RMI; distributed file systems; distributed application protocol design and examples; transactions; concurrency control and recovery, distributed transactions, nested transactions. Replication, fault tolerance; primary-backup and state machine approaches, models of distributed computation; vector clocks; reliable broadcast. Prerequisite: CS 520.

CS 550 Computer Organization and Programming
This course provides an intensive introduction to material on computer organization and assembly language programming required for entrance into the graduate program in Computer Science or Computer Engineering. The topics covered are: structure of stored program computers; linking and loading; assembly language programming, with an emphasis on translation of high-level language constructs; data representation and arithmetic algorithms; basics of logic design; processor design: data path, hardwired control and microprogrammed control. Students will be given assembly language programming assignments on a regular basis. No graduate credit for students in Computer Science or Computer Engineering. Cross-listed with CpE 550.

CS 551 Software Engineering and Practice I
(3-1-3)
Software design and development theory of software design, with emphasis on large systems. Models of the software process: specifications development, designing, coding and testing. Program abstraction with functional abstraction and with abstract data types. Top-down and bottom-up development methods. Common software architecture models. Specification validation, design verification, testing strategies, test coverage

issues. CS 551 and CS 540 may not both be taken. Prerequisite: CS 385.

CS 552 Software Engineering and Practice II
(3-1-3)
Covers the practical aspects of the software process. Cost and schedule estimation; management systems; technical vs. administrative responsibilities; documentation; libraries; configuration management; automated software tools; documentation. The class is partitioned into software teams, each of which develop and manage a typical large-scale software development process. Projects are drawn from on-campus computational needs. Prerequisite: CS 551.

CS 558 Computer Vision
An introduction to the field of Computer Vision, focusing on the underlying algorithmic, geometric and optic issues. The course starts with a brief overview of basic image processing topics (convolution, smoothing, edge detection). It then proceeds on various image analysis topics: binary images, moments-based shape analysis, Hough transform, image formation, depth and shape recovery, photometry, motion, classification, special topics. Prerequisite: CS 385 or CS 590 or equivalent; Corequisite: Ma 112 and Ma 115 or equivalent. Cross-listed with CpE 558.

CS 561 Database Management Systems I
Introduction to the use of relational database systems; the relational model; the entity-relationship model; translation of entity-relationship diagrams into relational schemes; relational algebra; SQL; normalization of relational schemes. Students who have had a previous course in database systems must obtain permission of the instructor to enroll in this course. Prerequisite: CS 590 or equivalent.

CS 562 Database Management Systems II
A continuation of CS 561. Review of the relational model, relational algebra, SQL and normalization; storage organization; indexes; B-trees and B+-trees, query optimization; concurrency control; recovery control. Prerequisite: CS 561 or equivalent.

CS 563 Object-Oriented Database Management Systems
Introduction to fundamental principles of object-oriented database management systems. Topics include: review of the relational database model and its deficiencies; concepts underlying object-oriented methodology; application of object-orientation to database management systems; transaction processing; object-oriented databases under client/server architectures. Prerequisite: CS 561.

CS 564 Software Requirements Acquisition and Analysis
Requirements Acquisition is one of the least understood and hardest phases in the development of software products, especially because requirements are often unclear in the minds of many or most stakeholders. This course deals with the identification of stakeholders, the elicitation and verification, with their participation, of the requirements for a new or to-be-extended software product. It deals further with the analysis and modeling of requirements, the first steps in the direction of software design. Finally, it deals with the quality assurance aspects of the software requirements phase of the software development process. This course is case-history and project oriented, and uses industry-standard software tools. Prerequisites: CS 540 or CS 551.

CS 565 Software Architecture and Component-Based Design
This course deals with the high-level (archi-

tectural) and low-level issues involved in the design of software systems/products. At the high level it deals with such issues as component-based design, cohesion, interconnection complexity and methods for minimizing the latter; it also deals with the use of middleware, performance analysis and simulation, and the use of COTS components. At the low level, it deals with object-oriented design, design patterns and code refactoring. Finally, it deals with validation and verification of both architecture and code designs. This course is case-history and project oriented. Prerequisites: CS 540 or CS 551.

CS 567 Software Testing, Quality Assurance and Maintenance
This course provides in-depth coverage of software testing, configuration management, quality assurance, and maintenance, both in terms of defect removal and enhancement. It deals with the performance of these activities in a variety of life-cycle models, from spiral-model or Rational Unified Process development, through lighter weight varieties of incremental and iterative models down to extreme programming. It is a project-oriented course. Prerequisites: CS 540 or CS 551.

CS 568-CS 569 Software Project I-II
These two courses, which must be taken in sequence, constitute the capstone of the MS program in Quantitative Software Engineering. Students will use knowledge and skills gained in earlier courses to build real software products, in teams, for real stakeholders. Prerequisites: For CS 568: CS 540, CS 533 CS 564, CS 565, CS 567, and the ability to program in C++, which may be obtained by taking CS 580 as an elective. (C++ is the basic programming language which will be assumed for all projects except in cases in which it is technically inappropriate.) Only in extraordinary cases will a student without all prerequisites be allowed to take this course, and then only with the written permission of the instructor. Prerequisite for CS 569: CS 568.

CS 570 Introduction to Programming in C++
An elementary introduction to the standard data types and programming constructs of C++. Students will be given weekly programming assignments. Not for graduate or undergraduate credit in Computer Science, Computer Engineering and Information Systems degree programs.

CS 571 Java
The course consists of an in-depth discussion of Java language and programming techniques. Comparison of Java to other languages, such as C/C++, is made throughout the course to emphasize various shortcomings of the language and their implications on design paradigms. Some aspects of GUI libraries, multi-threading support and Java native interface are also discussed.

CS 573 Fundamentals of CyberSecurity
This course studies the mathematical models for computer security (Bell-LaPadula, Clark-Wilson, Biba and Gligor models). It analyzes and compares, with respect to formal and pragmatic criteria, the properties of various models for hardware, software and database security. Topics also include: formal specification and verification of security properties, operating system security, trust management, multi-level security, security labeling, security auditing and intrusion detection, security policy, safeguards and countermeasures, risk mitigation, covert channels, identification and authentication, password schemes, access control lists and data fusion techniques. The course includes a project. Prerequisite: CS 520 or equivalent.

CS 580 The Logic of Program Design
Introduction to the rigorous design of functional and procedural programs in modern language (C++). The main theme is that programs can be reliably designed, proven and refined if one pays careful attention to their underlying logic, and the emphasis of this course is on the logical evolution of programs from specifications. Programs are developed in the UNIX environment. No graduate credit for students in Computer Science or Computer Engineering. The necessary background in logic, program syntax and UNIX is developed as needed, though at a fast pace. \Corequisite: Ma 502. Cross-listed with CpE 580.

CS 590 Introduction to Data Structures and Algorithms
Introduction to the design and analysis of algorithms. Standard problems and data structures are studied, as well as learning how to analyze the worst case asymptotic running time of algorithms. Students will be given programming assignments on a regular basis. No graduate credit for students in Computer Science. Prerequisite: CS 580 and Ma 502. Cross-listed with CpE 590.

CS 600 Analysis of Algorithms
The complexity and correctness of algorithms: big oh, big omega and big theta notations, recurrence relations and their solutions. Worst, average and amortized analysis of algorithms with examples. Basic and advanced data structures for searching, sorting, compression and graph algorithms. Students will be given programming assignments on a regular basis. Prerequisite: CS 590. Cross-listed with CpE 600.

CS 601 Algorithmic Complexity
Analysis of algorithms: resource-bounded computation, time and space complexity. Various models of computation will be studied. Complexity classes and reducibilities, hardness and completeness. Randomized algorithms and approximation algorithms. Prerequisite: CS 600.

CS 617 Elements of Compiler Design II
Optimization: dataflow analysis, copy propagation, dead code elimination, common subexpression elimination, code hoisting, elimination of redundant induction variables, compiling functional and object-oriented languages; control-flow analysis, static single-assignment form intermediate languages. Additional topics at discretion of the instructor. Prerequisite: CS 516.

CS 619 E-Commerce Technologies
The course provides an understanding of electronic commerce and related architectures, protocols and technologies. It describes the e-commerce concept, objectives and market drivers, as well as its requirements and underpinning techniques and technologies, including the Internet, WWW, multimedia, intelligent agents, client-server and data mining. Security in e-commerce is addressed, including types of security attacks, security mechanisms, Virtual Private Networks (VPNs), firewalls, Intranets and extranets. Implementation issues in e-commerce, including the design and management of its infrastructure and applications (ERP, CRM, SCM), are discussed. M-commerce is addressed; electronic payment systems with their associated protocols are described, and various B2C and B2B applications are presented. Also, policy and regulatory issues in e-commerce are discussed. Cross-listed with TM 619, CpE 619 and NIS 619. Prerequisite: CS 666, CpE 678, TM 610 or Mgt 776.

CS 625 Foundations of Distributed Computing
Design and analysis of distributed algorithms, and impossibility results showing when some problems are unsolvable. Models of synchronous and asyschronous distributed computing. Fault models including crash failures and malicious failures, and communication models including message passing and shared memory systems. Distributed algorithms and impossibility results for problems such as consensus, Byzantine agreement, clock synchronization, mutual exclusion, and secure multiparty computation. Prerequisite: CS 600

CS 630 Automata and Formal Languages
Analysis of finite automata and regular sets. Formal languages and grammars, Chomsky-hierarchy. Context-free languages and PDAs. Applications to parsing. Prerequisite: Ma 502 or equivalent.

CS 634 Decidability and Computability
Computable functions and Turing machines. Primitive recursive functions, recursive functions, loop-programs, while-programs. Decidability and solvability, unsolvable problems. High complexity classes and reducibilities, hardness and completeness. The arithmetical hierarchy and definability. The connection to complexity theory is emphasized throughout the course. Prerequisite: Ma 502 or equivalent.

CS 636 Integrated Services – Multimedia
Types of multimedia information: voice, data video facsimile, graphics and their characterization; modeling techniques to represent multimedia information; analysis and comparative performances of different models; detection techniques for multimedia signals; specification of multimedia representation based on service requirements; evaluation of different multimedia representations to satisfy user applications and for generating test scenarios for standardization. Prerequisite: EE 605, NIS 605 or CS 505. Cross-listed with CpE 636 and NIS 636.

CS 638 Interactive Computer Graphics II
Mathematical foundations and algorithms for advanced computer graphics. Topics include 3D modeling, texture mapping, curves and surfaces, physics-based modeling, visualization. Special attention will be paid to surfaces and shapes. The class will consist of lectures and discussion on research papers assigned for reading. In class, we will study the theoretical foundations and algorithmic issues. In programming assignments we will use OpenGL as the particular API for writing graphics programs. C/C++ programming skills are essential for this course. Prerequisites: CS 437, CS 537 or equivalent. Cross-listed with CpE 638.

CS 643 Formal Verification of Software
Formal systems for specification and verification of software; review of the first-order predicate calculus; abstract data types, formal specification, preconditions, postconditions, invariants, predicate transformers, proofs of correctness, partial and total correctness; correctness for assignments, alternatives, iterations, procedure calls. Prerequisite: CS 600.

CS 651 Introduction to Network and Graph Theory
Introduction to the theory and applications of networks and graphs. Topics include paths, connectivity, trees, cycles, planarity, network flows, matchings, color-

ings and some extremal problems. Prerequisite: Ma 502 or equivalent.

CS 655 Queuing Systems with Computer Applications I
Queuing models will be developed and applied to current problems in telecommunication networks and performance analysis of computer systems. Topics include elementary queuing theory, birth-death processes, open and closed networks of queues, priority queues, conservation laws, models for time-shared computer systems and computer communication networks. Prerequisite: EE 605, NIS 605 or CS 505. Cross-listed with CpE 655 and NIS 655.

CS 656 Queuing Systems with Computer Applications II
This course is a continuation of CS 655. Prerequisite: CS 655, CpE 655 or NIS 655. Cross-listed with CpE 656 and NIS 656.

CS 660 Graph Algorithms
Basic graph-theoretic notions; data structures for graph representation; running time analysis; review of depth-first search, breadth-first search and minimum spanning trees; network flow problems; graph connectivity; matchings; Eulerian graphs and digraphs; de Bruijn graphs; Hamiltonian graphs; traveling salesman problem; planar graphs; planarity testing; vertex and edge colorings; chromatic polynomials; five-coloring algorithm; the Four-color problem. Prerequisites: Ma 502 or equivalent and CS 600.

CS 666 Information Networks I
Introduction to information networks, architecture, communication models. Protocol definition for distributed networks including X.25 and SNA and performance analysis of various layers of protocols. Local area networks (LANs): CSMA/CD; token bus and token ring technologies and performance analysis of LANs. Routing and flow control techniques. Prerequisite: Graduate: CS 505; Undergraduate: Ma 222. Cross-listed with NIS 678 and CpE 678.

CS 667 Information Networks II
Advanced network architectures including integrated digital networks and Integrated Services Digital Networks (ISDN); narrowband and broadband ISDNs. Architectural design based on topological considerations, bandwidth assignment and connection management for services, flow control and routing designs. Satellite communications, multimedia services and communication techniques, ATM, SONET and SDH. Prerequisite: CS 666. Cross-listed with CpE 679 and NIS 679.

CS 668 Foundations of Cryptography
This course provides a broad introduction to cornerstones of security (authenticity, confidentiality, message integrity and non-repudiation) and the mechanisms to achieve them as well as the underlying mathematical basics. Topics include: block and stream ciphers, public-key systems, key management, certificates, public-key infrastructure (PKI), digital signature, non-repudiation, and message authentication. Various security standards and protocols such as DES, AES, PGP and Kerberos, are studied. Prerequisite: Ma/CS 503, CS 590; or permission of the instructor. Cross-listed with CpE 668

CS 669 Network Management
Hierarchical network management for LAN and distributed discrete and integrated services networks; network management concepts; administrative and operational management; performance management; fault management; maintenance management; security management and

architectural management of different ownerships. Concept of managed objects, manager-agent relationship, applications of network management protocols. Standard management protocols: SNMP and CMIP. Prerequisite: CS 666.

CS 670 Information Theory and Coding
An introduction to information theory methods used in the analysis and design of communication systems. Typical topics include: entropy, relative entropy and mutual information; the asymptotic equipartition property; entropy rates of a stochastic process; data compression; Kolmogorov complexity; channel capacity; differential entropy; the Gaussian channel; maximum entropy and mutual information; rate distortion theory; network information theory; algebraic codes. Prerequisite: CS 505.

CS 674 Theory of Object-Oriented Software Design
Theory of object-oriented design, classes, interfaces, inheritance hierarchy, correctness; abstract data types, encapsulation, formal specification with preconditions, postconditions and invariants, proofs of correctness; object-oriented software, objects and classes, genericity, inheritance, polymorphism, overloading; single and multiple inheritance, programming by contract, subclassing as subcontract, specification and verification; programming language examples include C++, Java, Smalltalk and Eiffel. Prerequisite: CS 600.

CS 687 Engineering of Large Software Systems
Students will learn how to deal with issues impacting industrial software developments. A broad range of topics will be covered emphasizing large project issues. Large software projects are those employing 50 or more software developers for three years or more. Throughout the course, emphasis will be placed on quantitative evaluation of alternatives. Specific examples and case histories from real projects in the telephone industry are provided. Students will learn how to create architectures for large systems based on the '4+1' model; how to use modern software connector technology; module decomposition; scaling of agile methods to large projects, the use of work flows to drive software process and database designs, test plans and implementation; configuration control and software manufacturing. The special issues of database conversion data consistency, database maintenance and performance tuning will be addressed for large data bases. The physical environment of the computer systems including multisite deployment; software releases and special management report generation are examined. Prerequisite: CS 540

CS 689 Software Reliability Engineering
Students will learn how to analyze, predict, design, and engineer the required and expected reliability of software systems. Case studies will be used throughout, including studies of sysems that worked well and of systems that failed in some crucial aspect. Examples of the types of systems which will be studied are the London Ambulance Dispatch System, the Lucent Telephone Switching Systems and the Mars and Voyager missions. Prerequisite: CS 533

CS 693 Cryptographic Protocols
This course covers the design and analysis of security protocols, and studies different attacks and defenses against them. Topics include: signature and authentication protocols, privacy, digital rights management, security protocols for wired, wireless and

distributed networks, electronic voting, payment and micropayment protocols, anonymity, broadcast encryption and traitor tracing, quantum cryptography and visual cryptography. The course includes a project. Prerequisite: CS 668

CS 694 E-business Security and Information Assurance
This course addresses the security of e-business and cyber environments from an end-to-end perspective, including data center security and access security. The information security phases of inspection, protection, detection, reaction and reflection are emphasized. Topics also include: server and application security, virtual local area networks (VLANs), secure access and financial transaction techniques, backup and disaster recovery techniques. The course also reviews financial Electronic Data Interchange (EDI) and smart card security in banking applications, and describes how the business and financial risks associated with security are estimated and managed. The course includes a project and related lab experiments. Prerequisite: CS 666 or TM 610 or equivalent. Cross-listed with TM 694

CS 700 Formal Semantics of Programming Languages
Methods for giving meaning to programming language constructs; Operational, Denotational and Axiomatic semantics. Introduction to algebraic tools; recursive definitions, fixed-point semantics; proving program correctness, program equivalence. Prerequisite: CS 630.

CS 765 Selected Topics in Computer Science*
A participating seminar on topics of current interest and importance in computer science.

CS 800 Special Problems in Computer Science (M.S.)*
An investigation of a current research topic at the pre-master's level, under the direction of a faculty member. A written report is required, which should have the substance of a publishable article. Students with no practical experience who do not write a master's thesis are invited to take advantage of this experience. One to six credits for the degree of Master of Science (Computer Science).

CS 801 Special Problems in Computer Science (Ph.D.)*
An investigation of a current research topic beyond that of CS 800 level, under the direction of a faculty member. A written report is required, which should have importance in Computer Science and should have the substance of a publishable article. This course is open to students who intend to be doctoral candidates and wish to explore an area that is different from the doctoral research topic. One to six credits for the degree of Doctor of Philosophy.

CS 900 Thesis in Computer Science (M.S.)*
A thesis of significance to be filed in libraries, demonstrating competence in a research area of Computer Science. Five to ten credits with departmental approval for the degree of Master of Science (Computer Science).

CS 960 Research in Computer Science (Ph.D.)
Original research of a significant character carried out under the guidance of a member of the departmental faculty, which may serve as the basis for the dissertation, is required for the degree of Doctor of Philosophy. Credits to be arranged.

*by request