Self-Study and Development Plan

Department of Computer Science
University of San Francisco

2130 Fulton Street, San Francisco, CA, 94117
tel: 415.422.6530, fax: 415.422.5800

March 12, 2008

Contents

1 Introduction

The Department of Computer of Science at the University of San Francisco offers rigorous undergraduate and graduate programs that prepare students for computing careers as well as for advanced study in computer science. In addition, the CS faculty and students are engaged in basic research, industrial and entrepreneurial efforts, as well as community outreach initiatives. Consequently, our students not only receive a foundation in computer science principles, but they are exposed to a wide range of *extracurricular* experiences. Our mission is to provide our students with currently accepted foundations while leveraging the specialities of our faculty and local resources.

In the last 10 years the department has undergone significant changes and improvements to the core undergraduate programs as well as changes in graduate program with the addition of a new degree, the Master of Science in Internet Engineering (MSIE). While we are pleased with our program and the balance we have achieved, we are always looking to ensure our students receive the best computer science educations and one that reflects current practice and expectations in industry and graduate programs.

We currently have 10 active tenure-track full-time faculty in the Department of Computer Science. Three of the faculty were hired within the last 6 years. Just last year we hired our first tenure-track female faculty member. Most of the faculty have active research programs and some are involved in community service efforts, open source software development, or entrepreneurship. Our students are exposed to a wide range of faculty activity.

The CS faculty held a full day retreat in the Fall 2007. The retreat helped us look at our accomplishments and areas that need improvement. The retreat conversations served as the basis for many issues raised in this self study.

The remaining sections of this introduction highlight distinguishing features of the USF CS Department, raises important ongoing questions that we want to address, and presents our institutional needs. Finally, we provide testimonials from students and our students' supervisors from academia and industry.

1.1 Key Aspects

This document provides extensive details on our programs, faculty, and students. Here are some of the key features of the Department of Computer Science:

- **Rigorous Programs** We offer rigorous undergraduate and graduate degrees that have an emphasis on software development. Most of our students end up with careers in software development, the ones

who want to continue on with their studies are well prepared. Most of our courses are programming intensive and we emphasize learning by doing.

- **Tight-knit Community** The department fosters a friendly, collaborative environment. Students develop strong ties with their peers and with the faculty. The Student ACM group runs a weekly social, pizza nights, and other activities that are well attended by students and faculty. Most faculty have an open door policy with respect to students and other faculty. We greatly value the high degree of interaction we have on the 5th floor of the Harney Science Center. In 2004 the CS department ran the first every FlashMob Computing event to create an instant volunteer supercomputer. This required tremendous effort from the entire department.

- **Project Courses and Student Research** Both our undergraduate program and graduate programs require at least one capstone project course. This course allows students to work on a significant software development or research project. Students work with outside industrial sponsors or with faculty advisors. The projects are usually done in groups. We have found that this less structured course gives students an opportunity to learn valuable organization and communication skills while producing something that they can showcase to future employers or publish in academic conferences. Many of our faculty make it a priority to involve undergraduates in their research. Students and faculty showcase their work at our annual CS Night event.

- **The Kudlick Computer Science Teaching Laboratory** A generous give from a USF CS Alumnus allow us to design and build a high-tech studio classroom. We teach most of our courses in the Kudlick classroom. The design of the room facilitates integrated lecture/lab learning. Students, especially in the introductory classes, get a chance to do real programming during lecture. This classroom has transformed the way we teach at the introductory level. We believe students learn faster with this mode of teaching.

- **Parallel Computing and Cluster Resources** With our local expertise in parallel computing we have been successful at exposing our students to real parallel systems. We were one of the first departments to require an undergraduate parallel computing course at the sophomore level. Undergraduates and graduates learn about parallelism by using two clusters: a 48 processor Infiniband-based cluster and a 48 processor Myrinet-based cluster. Many of our students have landed internships at Lawrence Livermore National Laboratory based on their parallel computing experience at USF.

- **Community Connections** The department has established an organization that matches CS students with local and international service learning projects, allowing our students utilize their technical knowledge for the service of others. A majority of our students work each week in inner-city community computer centers such as Network Ministries. We have a very successful service learning class in the undergraduate major and we run an annual trip to Tacna, Peru in which our students and faculty help install and maintain hardware and teach classes at two K-12 schools.

1.2 Key Issues

While we are interested in receiving feedback on all aspects of our programs from the outside evaluators, we have identified some key issues on which we are particularly interested in hearing feedback. These issues are:

- **Enrollments** We have experienced low enrollments in our undergraduate major beginning in 2003, like many CS programs across the nation. For the last 3 years we have graduated approximately 4 seniors a year. We've seen a slight increase in our freshmen courses, but we experience high attrition. Most students who leave either had a greater interest in something else such as graphic design or they find the major too difficult. Our low enrollments have required us to run required courses less frequently and to offer fewer elective options in a given year.

  Fortunately, our graduate programs are quite strong in terms of enrollment. We are able to place many graduate students in our upper division courses to fulfill deficiencies. This compensates for our high undergraduate attrition rate.

- **Number of Units for the CS Major** The undergraduate degree is 70 units out of 128 total required to graduate at USF. As such the CS major is one of the largest in terms of required units on campus. We are happy with the content and breadth our students receive and we hear positive feedback from our alumni and their employers and advisors. However, we recognize that requiring such a large number of units may have negative effects on recruitment, retention, and the possibility for our students to explore other fields of study, such as other sciences or humanities or to take advantage of college experiences such as spending a semester abroad.

- **Alternate CS Programs** Related to the unit issue is the possibility to attract and retain more students by introducing more interdisciplinary programs, ones that will ultimately require fewer units and may offer an unique experience for students who do not want a full CS major. We have come up with several proposals, but up to now have stopped short because of the added administrative overhead

in running such programs and the concern that while such programs may attract new students, they may also attract students away from the standard CS major.

- **Curriculum** Our curriculum has evolved to leverage our local expertise and to give our students a strong foundation in software development. We are not accredited with CSAB and the department feels that doing so will diminish many of the unique aspects of our degree and place too much of a burden on our students. Furthermore, we have revisited CSAB accreditation several times over the last 10 years and we find it simply too difficult to meet the requirements given the large CORE requirement at USF (44 units) and our four unit course model.

  We are dedicated to ensuring our students are exposed to fundamentals and emerging technology. We welcome outside views on how to best serve CS students through our required curriculum.

1.3 Key Needs

Based on our goals and the future success of our programs we have identified the following key needs for the department:

- **Attract more undergraduates** We need to work with other University units to help publicize our program. This may entail changing our curriculum to attract more students.
- **An additional full-time tenure track line** A new line would allow us to teach more sections of our core CS service course for non-majors. A new line would also help us us meet the demand we have in the graduate programs.
- **An additional systems administrator** We currently have one full time administrator and one half-time administrator. Ideally we would have one more junior-level system administrator. This will become more important as we expand our facilities in the construction of the new science building.
- **Enhance the University Scholar Program** USF has a University Scholar program to attract high achieving high-school students. We believe the program should be expanded and the funding amount should be restored to the old level of a 75% tuition reduction each year rather than a fixed amount for each year. Such students greatly enhance the learning environment for *all* students in a class.

1.4 Testimonials

In preparation for this self study we queried recent students, a graduate advisor, and a CTO of a local company to reflect on their experience with USF Computer Science Programs. Here are some of the responses we received.

The B.S. in computer science I received from USF provided me with the solid background necessary to succeed in my ongoing doctoral studies at University of California at Davis and my summer research appointments at Texas A&M University and Argonne National Laboratory. USF offers an excellent learning environment and a well-designed CS curriculum that allowed me to easily fulfill the undergraduate proficiency requirements at a Ph.D. granting institution. The professors are always willing to help the students, even outside of the scheduled hours, and to guide the students in independent study or research opportunities. I also found it rewarding to take advantage of such excellent opportunities as teaching assistantships and community service through the CS department's Community Connections project. All in all, my experience at USF was most memorable and rewarding. It has prepared me to succeed, and for that I will be forever grateful.

[Name Redacted] **, USF BSCS Class of 2004, Currently a Ph.D candidate in CS at UC Davis**

Its my great pleasure to write this assessment of Miss Anna Tikhonova, who is presently a PhD student at UC Davis under my supervision. My research area is in visualization. Anna came to UCD in September 2005. After she took my graduate-level visualization class in Winter 2006, she asked to join my group and I gladly accepted her. Before coming to UCD, apparently Anna has received a nice blend of training in computing. She did very well in my class, which required her to do a pretty ambitious research-oriented project. She demonstrated her problem solving ability as well as how quickly she can improve her other skills with the feedback that she received from me and the whole class. She is very thorough and do not accept any imperfection. In addition, we often find a touch of art in her visualization design, which makes her work stand out. I am very pleased with her performance and enjoy working with her. I believe she will be able to pursue a successful research-oriented career.

Professor Kwan-Liu Ma, [Name Redacted] **'s Ph.D. Advisor at UC Davis.**

Deciding to obtain my Masters in Computer Science at USF was one of the best decisions of my life. The small class sizes allowed me to closely interact with my professors, which lead to research opportunities outside of the standard courses. The faculty were open to new and different research ideas, which allowed me to pursue my thesis topic of tagging within the blogosphere. While working on my thesis, I also took graduate courses that covered topics that were applicable to current industry trends, such as distributed software development. These courses, in conjunction with my research and thesis work, helped me obtain a job at Yahoo!. I received positive feedback on my resume during the interview process at Yahoo!, because of my unique research experience. After I obtained the position, I began to see how my graduate education at USF gave me a real advantage in selecting a competitive position in software engineering.

[Name Redacted] **, USF MSCS 2007, Systems Software Engineer at Yahoo!**

Over the past several years, we have hired 5 graduates (BS and MS) from USF's computer science program. In addition, we have actively recruited interns from USF. The USF students have quickly matured into significant contributors at Tealeaf. We have been impressed by their aptitude, their background and their ability to succeed and flourish in our environment.

Robert Wenig, CTO Tealeaf Technologies, Inc.

2 History

Here is a summary of the major events in the Department of Computer Science:

- 1970 Department founded by Professor Jim Haag.
- 1971 Special Lecture Series started.
- 1981 Graduate program started (Masters of Science in Computer Science).
- 1994 Beginning of hiring of five new faculty members over the next decade.
- 1995 Software development emphasis instituted.
- 2000 Keck Cluster added to department resources, beginning of parallel processing emphasis.
- 2002 Kudlick Computer Science Teaching Laboratory opens.
- 2003 Masters of Science in Internet Engineering offered.
- 2004 Department's emphasis on public service projects begins, Community Connections, FlashMob Computing.
- 2007 Update to Keck Cluster, added two new clusters.

2.1 Beginnings

The Computer Science major first appeared in the General Catalog of the University of San Francisco in 1968–69, with courses offered by the Department of Mathematics. USF was among the first sixty universities and colleges to offer a degree in computer science, and one of the first private universities. A separate Department of Computer Science was first listed in the 1970–71 catalog. From the beginning, the department has carefully controlled and reviewed the curriculum to make it ACM compliant but distinctive in a local market of big-name universities. Although we held CSAB/CSAC accreditation in 1987–94, we eventually chose to develop our program in less restrictive ways. Characteristics to be noted are emphases on software development, knowledge of low-level as well as high-level programming, parallel computing, rigorous architecture and operating system experience, and capstone team software projects for all students. These projects, begun in 1994 and based on the model developed by Sacramento State University, involved research, development, and delivery of a tested system for on- and off-campus clients, both academic and industrial.

The department acquired a dedicated office and full-time secretary in 1989–90. Although still limited by Yeshiva Bar requirements not to have managerial responsibilities, CS department chairs began to have more control of budgets, expenditures, and service personnel. After more than a decade of machine support provided by students, we hired our first systems administrator in 2000; with the advent of the Keck Cluster, we gained a second administrator for the College with 50% time dedicated to the CS department.

2.2 Graduate Programs

The graduate program began in 1981, offering a Master of Science in Computer Science (MSCS). We have added an additional graduate degree, the Master of Science in Internet Engineering (MSIE), and serve students with three additional options, a 4+1-year BS+MSIE option, an Entrepreneurship emphasis, and a Bridge program for those with inadequate undergraduate preparation (typically, Bridge students come from foreign programs).

2.3 Computing Resources

Initially, the principal departmental computing resource was timeshare on a Univac 90/60 (an IBM 360 workalike mainframe) that was controlled by USF's IT department. The Department of Computer Science took delivery of its first machine, a Data General Eclipse MV/4, at the same time that the academic university acquired its first machine, a DG MV/10. Ironically, at the same time, administrative IT ran Vaxen, upgrading later to Alphas. The faculty had access to the better machines, but only under need-to-use access. The CS MV/4 was replaced by an IBM RS/6000 in 1990, which would be the last minicomputer/terminal installation. A few DOS microcomputers appeared in the CS department in 1983, followed by several Macs. Computers on faculty desks would not be seen campus-wide for four more years. Once university support for faculty computing became a reality, the CS department benefited from a general exception to narrow specifications. Six CS Silicon Graphics workstations were soon retired because commodity micros running Linux were better and more cost effective. Lab terminals were replaced with PCs and Macs. After years of being behind the curve with regard to hardware, the CS department finally enjoyed a steady replacement and upgrade policy for all of its boxes, including faculty/student research machines.

Without going into detail, a similar slow start with the Internet and campus networks has yielded to today's swift Internet connection, efficient fiber-based LANs, and spreading wireless. Student residences have offered port-per-pillow for many years now. The principal CS hands-on teaching activity has moved from the Harney 5th floor labs to the 2nd floor Michael D. Kudlick Computer Science Teaching Laboratory, which

includes 30 PCs available for lab work as well as podium-operated demonstration computers and multiple types of displays. The facility has allowed us to reconfigure many courses to include interspersed lab and lecture time.

2.4 Parallel Computing

The CS department has made a commitment to teaching parallel programming, requiring all majors to take the sophomore course CS 220 Introduction to Parallel Computing. In addition to access to parallel supercomputers at national labs, students have immediate access to the department's own cluster computers. Initially funded by the Keck Foundation, the department-engineered, department-built rack-and-switch Penguin Cluster supercomputer now offers 96 cores on an Infiniband interconnect. The cousin to Penguin is a 96 core cluster based on a slightly older Myrinet interconnect.

Peter Pacheco has written a foundational book on the Message-Passing Interface (MPI): *Parallel Programming With MPI*. He also led a group of several faculty members who have had scientific connections with Lawrence Livermore National Laboratory, including access to their supercomputers.

A graduate course in do-it-yourself parallel supercomputing led to the nationally hailed FlashMob Computing event on April 3, 2004. Over 700 fast microcomputers were brought to a campus gym by students, businesses, and community computing buffs. Volunteers vetted the machines, installed our special CD-based software, and wired them to network switches lent by Foundry Networks. Although the best complete Linpack was 77 Gflops on 150 machines, a peak rate of 180 Gflops was benchmarked on 256 simultaneous machines.

2.5 Special Lecture Series

In 1971–72, the Special Lecture Series in Computer Science (SLS) was initiated by George Ledin. It formalized into an ongoing weekly colloquium what previously had been ad hoc invited talks by computer specialists in the Bay Area. Over the last 36 years, more than 800 experts in the field have addressed students, faculty, and community members in this unique, required course that is nonetheless open to the general public.

2.6 Practical Courses

Building on a history of offering computing and computer literacy courses as a service to non-majors, we introduced practical one-unit courses in 1984. Now offered for two units per course, topics ranging from

Excel, Photoshop, and 3D Computer Graphics to Web 2.0 and beyond attract students of all majors to hands-on computing experience.

2.7 Community Connections

After some minor experiences with service learning in the early 1990s through a series of freshman seminars, in 2003 the CS department launched Community Connections, a multifaceted community service and service learning collaboration with San Francisco and with Tacna, Peru. The latter has involved annual trips to Peru by faculty and students and has been generously supported by CS alumni [Names Redacted]

. The fifth trip, planned for May 2008, will draw on a wider group within the university.

2.8 Faculty Patterns

This year we celebrate the appointment of our first female tenure- track faculty member, Sami Rollins. A number of women and minority members have taught part time throughout our history, but we have had only one full-time minority instructor.

After years of having only one professor with a Ph.D. in computer science, we now have seven. From as many as seven joint appointments with other science departments, we are down to four, in CS, Math, and Chemistry.

Our faculty is increasingly energetic about attending conferences, recruiting abroad, and giving guest lectures at Bay Area universities, colleges, and research labs.

Since 2005, faculty members have conducted the Summer Enrichment Program. This has attracted young women of ages 13-18 to USF for workshops and guest lectures aimed at exposing young women to the field of computer science.

2.9 Departmental Research and Development

In addition to ramping up publications in scholarly journals, our department has undertaken more relationships with industry and government. Starting with student placement and a sabbatical at Pixion, faculty members and students have been engaged recently with the Internet Archive, San Francisco city government, LucasArts, ANTLR Project, Lawrence Livermore National Laboratory, and most recently, David Wolber's own Web 2.0 startup, Personos. Our Sixth Annual CS Research Night in December 2007 hosted three faculty research talks and seventeen posters, more than ever before.

Joint research involving different blends of faculty, undergraduates, and graduate students has flourished in the last decade. Numerous papers have been accepted by prestigious conferences and refereed journals. Collaboration with colleagues at other universities has also increased.

2.10 Founders

We recognize and honor three men who founded the Department of Computer Science and set its direction for many years. James N. Haag came to USF in 1965 with a doctorate from UC Berkeley. He was charged with starting a computing program from a few courses that had been offered in the Department of Mathematics. George Ledin joined him from UC Berkeley in 1965 as a computer science lecturer and probably the second CS faculty member, although a separate CS department would not emerge from Mathematics for five more years. Ledin would later initiate both SLS and the graduate program. Michael Kudlick, with a Ph.D. in applied math from MIT, arrived from SRI in 1974. All would be chair of the department several times before they left for another position or retired. Alfred S. Chuang, Cofounder/CEO/Chairman of BEA Systems, was so positively influenced by his contact with Kudlick that he donated $2.5M to USF to build and maintain our excellent and innovative classroom-laboratory named after and dedicated by Professor Kudlick.

3 Undergraduate Curriculum

3.1 Overview

The Computer Science major is designed to prepare students for both careers in industry and graduate programs in Computer Science. The program is quite rigorous, requiring a large number of technically demanding courses. The high requirements for the major have paid off in terms of student outcomes – our students who have entered top-tier graduate programs in computer science have reported that they feel well prepared, and employers are satisfied by the quality of our graduates.

The Computer Science major consists of 70 required units: 12 units of mathematics, 8 units of science, and 50 units of computer science. In addition to the Computer Science requirements, students must fulfill the university requirement of 36 additional CORE general education requirements (the Computer Science requirements cover 8 of the 44 units required in the CORE) and 8 units of a foreign language, for a total of 114 of the 128 units required for graduation. Thus after completing all of the department and university requirements for a Computer Science degree, students are left with 14 units to be taken as free electives, effectively 4 additional classes. Note that students who perform poorly on the writing and mathematics placement tests may need to take up to 8 additional units in remedial coursework, for a total of 122 of 128 units, leaving just 6 additional units (effectively 2 additional classes). Note also that because of the pre-requisite structure of the classes required for the major, it is extremely difficult, if not impossible, to complete all of the major requirements in 3 years. Thus we cannot easily accommodate either transfer students or students who decide to pursue computer science after their freshman year.

Students must receive a grade of C or higher in all CS required courses (including Math and Science). This policy was instituted to ensure that students master the fundamentals before moving on to more advanced courses. If a student receives a grade lower than a C, the student is required to retake the course. We actively monitor student progress to help identify and advise poor performing students.

3.2 Program

3.2.1 Unique Features of the Major

Our computer science major has several unusual features which help us stand out as a department:

- **Lecture/Lab Format** All of our early classes are taught in an interactive lecture/lab format, where students get a quick introduction to a topic, then immediately get their fingers dirty doing examples

before continuing with more lecture. This intermixing of lecture with hands-on work allows students to learn conceptually difficult topics more quickly. This has been especially helpful with the use of Python in the intro classes, where students can get straight to implementing complex topics without too much syntax getting in the way.

- **Early Parallel Class** We require a parallel class quite early in the program, at the beginning of the Sophomore year. Writing parallel code requires a different kind of thinking from sequential coding, and the earlier students are exposed to parallel topics, the deeper their knowledge by the time they graduate. This course has already reaped rewards in terms of student performance in the operating systems class. Just as Object Oriented Design has moved from an advanced topic to an introductory one over the past 20 years, we believe that parallel classes should be taught earlier in computer science curricula.

- **Compilers/Architecture Link** In the architecture class, students build a complete pipelined processor from gates that implements a subset of the MIPS instruction set. In the compilers class, students write a complete compiler that compiles a subset of Java down to a subset of MIPS, which can be run on their processor in simulation.[1] Thus the students graduate with a complete understanding of the entire process of compiling and executing a program, from a high-level language all the way down to pushing bits on the wire.

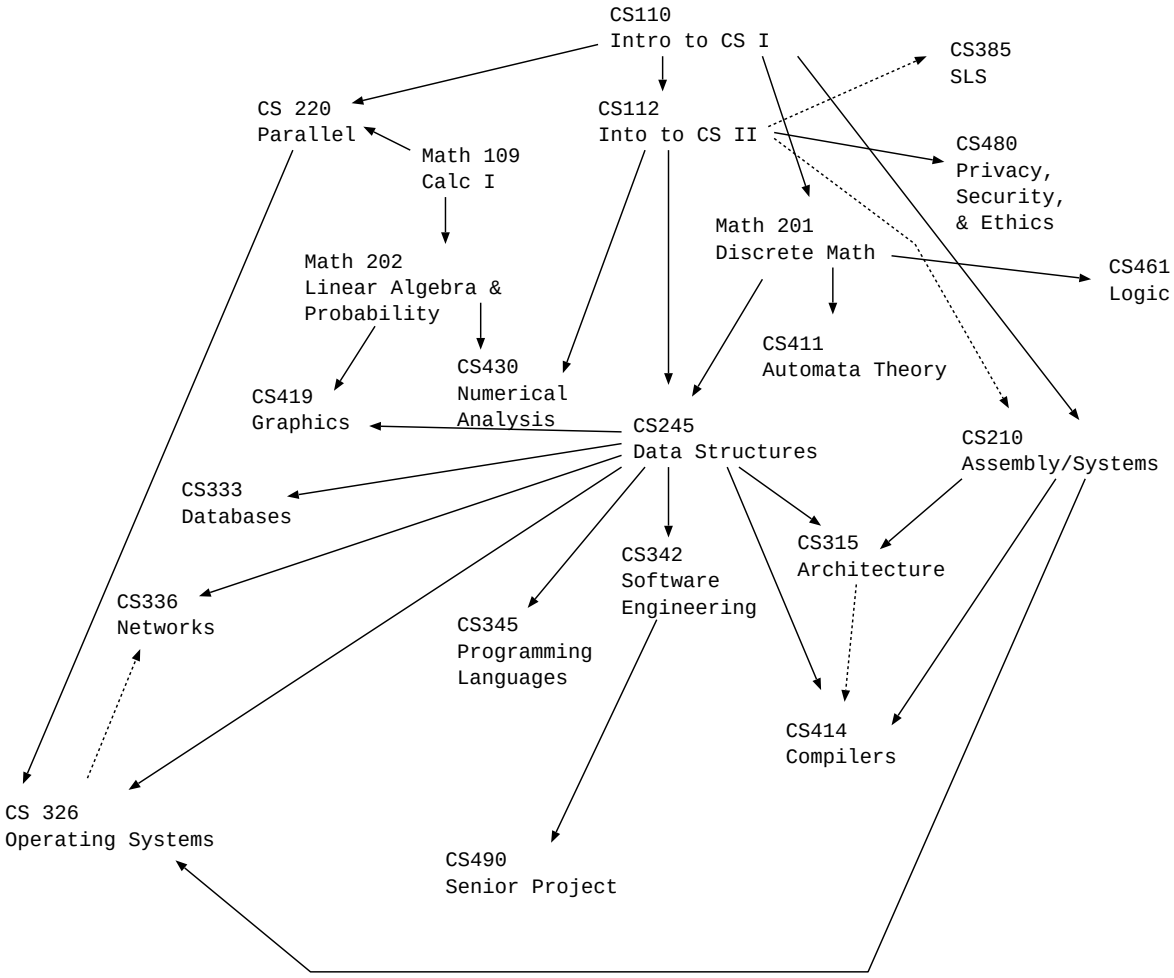
- **Capstone Project/Research Opportunities** All students are required to complete a capstone project, where students work either with an industry sponsor or a faculty member. Students working with industry sponsors get a complete picture of how software development is done in a real-world setting, while students opting to work with faculty members get an opportunity to work on a research project.

3.2.2 Computer Science Major

The Computer Science Major consists of 12 units of mathematics, 8 units of science, and 50 units of computer science. See the prerequisite dependancy graph below:

[1] The chip designed in architecture does not do multiplication, so the students need to do that in software, but that requires a trivial change to their code generator.

Computer Science Prerequsite Chart



Required Math and Science Courses

- Math 109 Introduction to Calculus and Analytic Geometry I (4 units)

  Differentiation of algebraic, exponential, logarithmic, trigonometric, and inverse trigonometric functions, and hyperbolic and inverse hyperbolic functions; implicit differentiation; curve sketching; indeterminate forms; velocity and acceleration; optimization; other applications of differentiation; Fundamental Theorem of Calculus, with applications to area and volume.

- Math 201 Discrete Mathematics (4 units)

  Prerequisite: CS 110. Topics include algebraic structures, graph theory, combinatorics, and symbolic logic. Though this class is in the mathematics department, it is designed for computer science students, and is often taught by faculty with dual appointments in mathematics and computer science.

- Math 202 Linear Algebra and Probability

  Matrix arithmetic and matrix algebra (determinants, adding and multiplying matrices, matrix inverse, using matrices to solve systems of equations), geometric applications of linear algebra (matrices as transformations, vectors in 2- and 3-dimensions, equations of planes, etc.); discrete probability, random variables, discrete and continuous probability distributions (including binomial and normal), expected value and variance. This class was also designed specifically for computer science students.

- Physics 110 General Physics I (4 units)

  This is the first course in the three-semester fundamental sequence of calculus-based physics. It is designed to meet the needs of the student whose major is Physics, Chemistry, Mathematics or Computer Science; in addition, it is recommended for Biology and Environmental Science majors who would prefer a stronger physics background than is provided by PHYS 100-101, Introduction to Newtonian mechanics of particles, systems of particles, and rigid bodies. Special topics include oscillations and gravitation. Three hours lecture and three hours lab weekly. Offered every Fall.

- Physics 213 Introduction to Electromagnetism and Electronics

  Prerequisite: PHYS - 110. This course is designed to meet the needs of the student whose major is Computer Science. Introduction to the physics of electricity and magnetism (including charge and current, electric and magnetic fields, and basic circuit theory) and digital electronics (including analog-digital converters, logical networks, flip-flops, shift registers, combinatorial logic, etc.). Three hours lecture and three hours lab weekly. Offered every Spring.

Required Computer Science Courses

- CS 110 Introduction to Computer Science I (4 units)

  No Prerequisite. Basic programming concepts, including use of functions & procedures, parameter passing, block structures, data types, arrays, abstract data structures, conditional control, iterative and recursive processes, and input/output in programming solutions to a variety of problems. Top-down and bottom-up design and functional decomposition to aid in the development of programs. These concepts are introduced using Python. The class is taught using an interactive lecture/lab combination, where short lectures are followed immediately by hands-on exercises using the Kudlick classroom. Offered every semester.

- CS 112 Introduction to Computer Science II (4 units)

  Prerequisite: CS 110. Design and development of significantly sized software using top-down design and bottom-up implementation. Dynamically allocated data, object-oriented programming, architecture of memory, basics of language translation, and basics of algorithm analysis. Development of simple graphical user interfaces. These concepts are introduced using Java. As with CS 110, The class is taught using an interactive lecture/lab combination, where short lectures are followed immediately by hands-on exercises using the Kudlick classroom. Offered every semester.

- CS 210 Assembly Language and System Programming (4 units)

  Prerequisite: CS 112. Introduction to machine structures, data representations, programming in assembly language; I/O programming and macros. Structure, design, and implementation of computer system software and utility programs. Students learn the x86 instruction set architecture. Four hours lecture. Offered every Fall.

- CS 220 Introduction to Parallel Computing (4 units)

  Prerequisites: Math 201 and CS 110 (B or better) or Math 109 and CS 112. Overview of parallel architectures. Programming shared and distributed memory parallel computers. Parallel program performance evaluations. Programming in C, using MPI, Pthreads, and OpenMP. Offered every Fall.

- CS 245 Data Structures and Algorithms (4 units)

  Prerequisites: CS 112 and Math 201. Algorithm analysis and asymptotic running time calculations. Algorithm design techniques and implementation details. Algorithms for sorting and searching, trees, graphs, and other selected topics. Four hours lecture. Medium-heavy programming course. Offered every Spring. This class is a prerequisite for almost all upper division courses, and should be offered every semester. Given student numbers, however, we have only been able to offer it once a year.

- CS 315 Computer Architecture (4 units)

  Prerequisites: CS 210 and CS 245. Performance analysis techniques, instruction set design, computer arithmetic, digital design, processor implementation, and memory systems. Performance enhancement using pipelining and cache memory. In this course, students build a complete pipeline processor from basic logic gates, which implements a subset of the MIPS instruction set architecture. Offered every Spring.

- CS 326 Operating Systems (4 units)

  Prerequisites: CS 210 and CS220 and CS 245. The design and implementation of operating systems. Study of processes, threads, scheduling, synchronization, interprocess communication, device drivers, memory management, and file systems. Students complete substantial programming projects. Offered every Fall.

- CS 342 Introduction to Software Engineering (4 units)

  Prerequisites: CS 245. Software process models and theory; structured development as a precursor to object-oriented development; advanced object-oriented design including modeling languages, design patterns, assertions, and dynamic binding and polymorphism; advanced programming techniques, including complex memory management and the design and use of (generic) components; software testing; user interface development; and client-server web development. A programming-heavy class in which students create large-scale programs, often in teams. Offered every Spring.

- CS 385 Special Lecture Series (1 unit * 2)

  No prerequisites. Weekly colloquium and discussion session on current developments in various aspects of computer science. Students may register for this course in more than one semester. Majors must take this course at least twice. One hour lecture. Offered Fall and Spring.

- CS 414 Compilers (4 units)

  Prerequisites: CS 210 and CS 245. Lexical analysis, parsing, semantic analysis, and code generation. Optimization techniques. Compiler design tools and compiler compilers. Students write a compiler that compiles a subset of Java to MIPS, which can be executed using SPIM, or on the chip they designed in Architecture. Offered every Spring.

- CS 490 Senior Team Project (4 units)

  Prerequisites: CS 342. Students working in teams investigate, specify, design, implement, test, document, and present to their classmates a significant software project. Sound software engineering practices are presented in lectures and used to evaluate each stage of the project. Written and verbal communication is emphasized through frequent documentation submissions, informal group discussions, code walkthroughs, and student presentations. With the instructor's permission, the course may be repeated for credit. Often students work on research projects with professors in this course. Prior to the 4-unit model, this course was a year-long two course sequence. Currently, students who want

to work the full year on a project can take a second semester as an independent study. Offered every Fall.

- Two Computer Science Electives, chosen from:
  - CS 333 Database Systems (4 units)

    Prerequisite: CS 245. Data modeling, record storage, and file organization; database theory; relational, hierarchical, and network models; database management systems and query languages, programming language interfaces to databases; web-based client-server development. We do not have any database people on faculty, and this course is almost never offered.
  - CS 336 Computer Networks (4 units)

    Prerequisite: CS 112 (CS 245 recommended) Current methods and practices in the use of computer networks to enable communication. Physical and architectural elements, and layered models of networks. Communication protocols and associated algorithms; local and wide area networks; network security. Four hours lecture. Offered every Fall.
  - CS 345 Programming Language Paradigms (4 units)

    Prerequisites: CS 112 (CS 245 recommended) Syntax, semantics, concepts, capabilities, and implementation details of several different programming languages, including imperative, functional, object oriented, and logical languages. Comparative advantages and disadvantages of different languages and paradigms. This class (which used to be required under the old 3-unit curriculum) is now rarely taught.
  - CS 411 Automata Theory (4 units)

    Prerequisites: Math 201 and Math 202. Finite state automata with bounded and unbounded memory. Regular languages and expressions. Context-free languages and grammars. Push-down automata and Turing machines. Undecidable languages. P versus NP problems and NP-completeness. This class has three major goals: Teaching students how to think formally and prove statements about computation, giving students experience using different kinds of computational formalisms, and understanding NP-Completeness. Offered every other fall.
  - CS 419 Computer Graphics (4 units)

    Prerequisites: CS 112 and CS210 and Math 109 and Math 202. Theory and production of interactive computer graphics. Topics chosen from graphics programming and algorithms, modeling, rendering, ray-tracing, and animation.

- CS 430 Numerical Analysis (4 units)

  Prerequisites: CS 112 and Math 109 and Math 202. Floating point representation of numbers, error analysis, root finding, interpolation, numerical integration and differentiation, numerical solution of linear systems, numerical solution of differential equations. Given low student interest, this class is rarely offered.

- CS 461 Logic for Computer Science and Math (4 units)

  Prerequisites: Math 201. Propositional and predicate calculus, syntax and semantics, formal theories, logic programming, lambda calculus. Applications of logic to computer science and mathematics. Four hours lecture. Offered every other year.

- CS 480 Computers and Society: Privacy, Security, and Ethics (4 units)

  Prerequisites: CS 112. Computer and network security measures; encryption protocols. Ethical theory and applications in computing. Seminar discussion on value systems, social impact, and human factors, and about use and misuse of computers. This course has a USF Service Learning designation, so our majors fulfill their service learning requirement by taking this course.

With permission from the department, students can also take graduate courses in computer science to fulfill the Computer Science Elective requirement

3.2.3 Computer Science Minor

The Computer Science Minor consists of 20 units of computer science courses, as follows:

- CS 110 Introduction to CS I
- CS 112 Introduction to CS II
- One of:
  - CS 245 Data Structures and Algorithms
  - CS 210 Assembly Language and Systems Programming
- Two additional computer science courses, as approved by the department. Note that Math 201, discrete mathematics (a prerequisite for CS 245, Data Structures and Algorithms) may be taken as one of these two electives.

While we have traditionally had relatively few minors in computer science, we are currently getting more interest in the minor from students in the school of business and graphic design.

3.2.4 Service Courses

The computer science department also offers the following service courses:

- CS 107. Computing, Robots, and Java (4 units)

  An introduction to computer science for non-majors with little prior programming experience. Students develop programs using visual and high-level programming languages to control robots, create animated simulations, and build Internet and general applications. In addition, students are exposed to an overview of computing and its influence on modern society. Offered Fall and Spring. This class counts for a CORE requirement for non-majors. It is very poplar, and is taught by full-time faculty. We could easily open another section of this course but we do not have enough faculty hours.

- 2-Unit practical computer science courses.

  These courses are designed to give non-majors specific, practical skills. They are all 2-unit courses, so that they can be easily added to a student's standard 16-unit schedule to get 18 units – which is the maximum number of units a full-time student can take without an increase in cost. They are typically *not* taught by regular faculty members, but rather part time staff.

  - Creating Images: Photoshop I
  - Word Processing
  - Creative Tools: Publishing I
  - Spreadsheet Computing I
  - Database Computing I
  - Web Site Design & Development I
  - Web Site Design & Development II
  - Presentations with Powerpoint
  - Animations with Flash

3.3 Admission

There is no special admission procedure for the computer science major. Any student admitted to the university is welcome to major in computer science. Students who show an interest in computer science

are encouraged to take the first course in the introductory sequence to determine if computer science is an appropriate major.

3.4 Advising

All freshman and sophomore students are required to see their advisor every semester before registering for courses. In their first meeting, the advisor and student work out a complete schedule of courses that will ensure timely graduation. Since the major requirements are relatively regimented, students usually follow a standard template. In subsequent advising sessions, the advisor monitors the student's progress towards graduation, making sure that they stay on track. While advising for most students is straightforward and advising appointments tend to be short, the regular contact with the advisor helps build more personal relationships. Regular advising also helps to identify problems early, so that they can be solved before becoming insurmountable.

A few years ago, the department decided to no longer require advising for juniors and seniors, though all students are welcome and encouraged to meet with advisors. The thought was that students already know exactly what courses they need, and signing off on student schedules became a rubber stamp process. Given the difficulty of the junior year, and one or two instances of students not taking an optimal set of courses late in their career, the department is considering requiring advising for all students before every semester.

In addition, the department secretary does an audit each semester, flagging all students who are in classes for which they do not have all of the required prerequisites. This list is sent to the department chair, who works with the professors of the classes to ensure that students do not stay in classes for which they are not prepared.

3.5 Academic Quality

We currently use a "ramp up" approach in the CS major. We introduce students to programming in CS 110 at a relatively moderate pace. This is necessary to accommodate a wide range in student preparation. Advanced students are encouraged to skip CS 110 and take CS 112. While the sophomore year builds on CS 110 and CS 112 with data structures, algorithms, assembly and parallel programming, the pace and rigor is slightly higher than the freshman year courses. Currently the junior year presents a huge increase in student expectations. It is well known that the junior year is the hardest and student who make it to the junior year are mentally prepared for these classes, which include operating systems, software engineering, and computer architecture.

As indicated previously, we require that CS majors earn a C or higher in all required courses for the major. This includes the Math and Physics courses. Generally speaking, the grading structure and course progression works well. The students who make it to the junior year are prepared to take the courses and can do well. Students that make it through the junior year almost always finish out the senior year to graduate. Most of our attrition occurs in the sophomore year. Students who leave the major typically sign up for the CS minor, but few of them complete the minor.

4 Graduate Curriculum

4.1 Overview

The University of San Francisco's graduate program in Computer Science offers Masters of Science degrees in Computer Science and in Internet Engineering. The degrees prepare students for careers in software development, research, or for entry into a Ph.D. program. The department balances excellence in teaching with active research programs. Masters students are encouraged to do research and to publish results with faculty. In addition, our Master's program has benefitted from close relationships with local industry.

The graduate program is intended for students with an undergraduate degree in computer science, engineering, or a related technical or scientific discipline. The program is also open to students with an undergraduate degree in other fields and extensive on-the-job experience in software development. Applicants are required to have coursework or work experience in object-oriented and low-level programming, calculus, linear algebra and discrete math. The program is focused on software development with the following objectives:

- Provide students with a broad background in software development and other core disciplines of computer science.
- Provide students with experience in the development of a substantial software project.
- Expose students to the areas of networking and concurrent computing.

The department offers two unique options: an entrepreneurship emphasis in conjunction with the highly ranked entrepreneurship program in the business school and a practical option that allows both international and American graduate students to satisfy an elective course by working in industry. In addition, the department offers a special degree for students applying from countries whose education system has a three-year computer science bachelor's degree. Students take a year of preparatory coursework before embarking on a Master's degree.

4.2 Demographics

Looking back over the past 10 years, our graduate programs have grown from 7 incoming students (1998) to 27 incoming students for the Fall 2007 semester. The number of applicants has grown threefold. The graduate program currently has about 60 students. We consistently have about 35% women and between

15% and 30% American students. The recent introduction of the entrepreneurship emphasis has seen an increase in the number of American students.

4.3 Program

The graduate program involves the completion of 36 units of coursework. Graduate students undertake 9 courses, or 8 courses and a master's thesis (each course is 4 units). Graduate students who have not met the Foundation Requirements may need to complete additional coursework at USF.

The plan of graduate study includes the following:

- 3 graduate courses in Software Development
- 1 graduate course in Concurrent Computing and Networking
- 2 graduate courses in Software/Hardware interface
- 2 elective courses
- 1 master's project course

A very few of our master's students choose the thesis option due to the perceived difficulty and potentially open-ended time commitment. Over the last ten years we've had 5 students successfully complete theses.

4.3.1 Emphasis in Entrepreneurship

The United States excels at entrepreneurship, and USF claims one of the top entrepreneurial business schools in the nation. Ranked in the Top Tier of University Entrepreneurship Programs by Entrepreneur magazine and in the Top 25 of "America's Most Entrepreneurial Campuses" by Forbes and Princeton Review, the M.S. in Computer Science Program has joined with USF's MBA Entrepreneurship Program to offer the classic M.S. in Computer Science with an emphasis in Entrepreneurship. Students enrolled in the entrepreneurship emphasis take a standard MSCS or MSIE degree, but take their electives in the Entrepreneurship Program from the business school alongside MBA students:

MBA664 Creativity and Innovation (1st Spring semester)

MBA662 Global Product Development (2nd Fall semester)

MBA661 Entrepreneurial Management (2nd Spring semester)

Students take 7 (4 unit) computer science courses and 3 (3 unit) MBA courses for a total of 28+9=37 units versus the 9 course (36 unit) pure MS computer science. In addition, students are required to prepare a

business plan and prototype during the master's project course and enter the USF International Business Plan Competition, one of the world's premier competitions.

4.3.2 Practicum Option

Computer science graduate students may elect to take a practicum option in order to gain practical experience in industry while working on their graduate degrees. Students enroll in the 2 unit (2 credit hour) CS 695 Practicum course during each semester in which they work (4 units during the summer). While students may work throughout their graduate program, they may only apply 4 units towards their 36 unit degree requirements. This option is reserved for exceptional international and American students. International students in F and J status must apply for the appropriate off-campus employment authorization in order to do the practicum.

4.3.3 MSCS Program for International Students

International students who have completed a three-year Bachelor's degree can apply for the M.S. in Computer Science Bridge Program. This program is designed for international students with undergraduate degrees in technical disciplines who need an additional year of preparatory courses to successfully pursue their Master's degree in Computer Science. As of Fall 2007, we have two students officially enrolled in this program.

4.3.4 Graduate Courses

- CS 601 Object-Oriented Software Development (4 units)

  Prerequisite: CS 245 and experience with an object-oriented programming language. A study of software development. Software engineering principles and structured methods are discussed as a prelude to the focus on object-oriented approaches. All phases of the software life cycle are covered, including analysis, design, implementation and testing, and maintenance. Other topics include user interface design and development, software reuse and the design of reusable software components, software patterns, and web-based client-server programming. Offered every Fall and Spring.

  This is one of the core courses of the graduate programs. We currently offer it every Fall and Spring by cross-listing it with CS 345, the undergraduate software engineering course. Graduate students are required to do more work and are held to higher grading standards.

- CS 615 Computer Architecture (4 units)

Prerequisites: CS 245, CS 315, CS 326. Survey of contemporary computer organizations covering early systems, instruction set design, processor implementation (pipelining, multiple issue, and speculative execution), memory hierarchy design (on-chip and off-chip caches, translation lookaside buffers, and virtual memory), input/output (devices, busses, and processor interfaces), performance evaluation, and current research topics. Project required.

This is course is rarely offered because we only have one faculty member qualified to teach the course.

- CS 620 Network Design (4 units)

  Prerequisites: CS 245, CS 326, CS 336. Overview of local and wide-area computer networks and contemporary lower-layer network protocols. Topics to be chosen from: switched networks, broadcast networks, multiplexing, layered protocol models, physical aspects of data transmission, data-link protocols, network modeling, performance issues, and current research in network design. Term paper or project required.

  This course is rarely offered because we find students prefer to learn higher-level network programming found in CS 621.

- CS 621 Network Programming (4 units)

  Prerequisites: CS 245, CS 326, CS 336. Network application programming. Upper-layer protocols and their interfaces. Topics to be chosen from: TCP/IP, sockets, remote procedure calls, network management, client/server programming, internet protocols (FTP, SMTP, HTTP, and SNMP), higher-level interoperability (CORBA), performance issues, and security. Project required. Offered every Spring.

- CS 625 Parallel and Distributed Computing (4 units)

  Prerequisites: CS 245, CS 315, CS 326. Introduction to shared- and distributed-memory architectures. Mechanisms for parallelism: locks, barriers, semaphores, monitors, message-passing, RPC, and active messages. Programming shared- and distributed-memory systems. Introduction to parallel algorithms and parallel performance prediction and measurement. Programming languages and libraries that support parallel and distributed computing. Offered every Spring.

- CS 630 Advanced Microcomputer Programming (4 units)

  Prerequisite: CS 210 or equivalent experience with Intel 80x86 Assembly Language. In-depth introduction to the protected-mode architecture of Pentium-family processors and supporting peripheral

controllers. Topics include memory segmentation and paging, privilege transitions, multitasking, exception handling, debugging, performance monitoring, system management mode, virtual-8086 emulation mode, and APIC inter-processor interrupts. Four hours lecture. Offered every other Fall.

- CS 635 Advanced Systems Programming (4 units)

  Prerequisite: Requires knowledge of C/C++ and acquaintance with UNIX/Linux operating systems. This course focuses on advanced hardware and software topics in systems programming, such as device-driver design, interprocess communication, and kernel-module programming in the Linux environment. Four hours lecture. Offered every other Fall.

- CS 636 Operating Systems (4 units)

  Prerequisites: CS 245, CS 315, CS 326. Study of the design and implementation of modern operating systems. Topics chosen from: operating system structure, scheduling, protection, virtual memory, communication mechanisms, concurrency, threads, multiprocessor support, distributed systems, performance evaluation, and current operating systems research. Project required. Offered every other Spring.

- CS 640 Computer Graphics (4 units)

  Prerequisites: CS 419 or all of MATH 202 , CS 245 , and knowledge of a graphics API. Concepts, algorithms, and mathematics of computer graphics in two and three dimensions. Homogeneous coordinates; modeling and viewing transformations. Geometric, procedural, and lighting models. Rendering, shading, and animation methods.

  This class is rarely offered because we do not have enough staffing.

- CS 652 Programming Languages (4 units)

  Prerequisites: CS 326 and either CS 345 or CS 414. Study of the design and implementation of software development languages. Topics chosen from: syntax, semantics, translation, run-time systems, advanced programming techniques, and debugging. Language families to be chosen include: functional, logic, visual, formal specification, design, pattern, database, and concurrent. Project required. Four hours lecture. Offered every other Spring.

- CS 662 Artificial Intelligence Programming (4 units)

  Prerequisite: CS 245. Use of artificial intelligence techniques to solve large scale problems. Search strategies, knowledge representation, and other topics chosen from: simulated annealing, constraint

satisfaction, logical and probabilistic reasoning, machine learning, expert systems, natural language processing, neural networks, genetic algorithms, and fuzzy logic. Both theoretical foundations and practical applications will be covered. Coursework includes written assignments and programming projects. Offered every Fall.

Most of our students will take this course and CS 601 in their first semester. Currently, the projects for CS 662 are done in Python.

- CS 673 Algorithms (4 units)

  Prerequisite: CS 245. Algorithm analysis and asymptotic running time estimates. Expected running times and amortized analysis. Design techniques, including divide and conquer, greedy, and dynamic programming. Algorithms for searching and sorting, graphs, and advanced topics. Four hours lecture. Offered every Spring.

- CS 675 Theory of Computation (4 units)

  Prerequisites: CS 245, CS 411. Topics to be chosen from: models of computation and formal languages, computability and complexity, P and NP completeness and P =? NP, advanced computing models.

  This course is rarely offered.

- CS 680 Internet Systems Research (4 units)

  Prerequisite: CS 662. Survey of Internet systems research including the anatomy of the web, search engine architecture and algorithms, information retrieval, crawling, text analysis, personalization and context, collaborative environments, and the semantic web.

  This course is required for the MSIE degree.

- CS 682 Distributed Software Development (4 units)

  Prerequisite: CS 601. Internet application development, including server-side technologies such as scripting languages, template frameworks, web page mining, and distributed computing issues such as peer-to-peer, multi-cast, and distributed agents.

- CS 684 Human-Computer Interaction (4 units)

  Prerequisite: CS 245. Design principles and techniques used to facilitate the interaction between people and computers. Topics covered include user-interface design and evaluation, web site design,

prototyping, usability engineering, presenting complex information, hypertext, multimedia, scientific visualization, input devices, ubiquitous computing, and cognitive models.

We have decided to drop this course from the MSIE degree and we will not offer it on a regular basis. Staffing this course has been difficult.

- CS 686 Special Topics in Computer Science (1-4 units)

  Topics not covered by other CS curricular offerings. Students may register for this class in more than one semester. Consent of instructor required. Offered intermittently.

  We have taught a number of interesting special topics courses. Here is a list of planned and recent offerings:

  - Wireless Sensor Networks (Fall 2008)
  - Gigabit Ethernet Programming
  - Intel EM64T and VT Programming
  - Network Device Drivers
  - Implementing Concurrent Programming Languages

- CS 687 Digital Society (4 units)

  A study of the effects of computing and the Internet on modern society. Topics include digital libraries, e-commerce, copyright law and open source movements, on-line communities, education and technology, and privacy and security.

  This course is rarely offered.

- CS 689 Residency in Internet Engineering (4 units)

  Prerequisites: CS 680 and CS 682. Participation in a cooperative work program with one of the USF affiliated organizations. Typically, students will work in groups and be supervised jointly by both an affiliate manager and a USF professor. Work is usually done in the summer.

- CS 690 Master's Project (4 units)

  Prerequisite: CS 601 and Regular Status. At the discretion of the instructor, the project will be either a sponsored project for a commercial concern or or other institution or a research project. In either case, the project will result in the specification, design, and development of a significant software system

with full documentation, an oral presentation to the university community, and a written report. Four hours lecture. Offered every semester.

- CS 695 Practicum Study (2 units)

  Prerequisite: CS 601 and Practicum Option status. Participation in a supervised work program where students apply USF coursework knowledge in a practical setting. Work is supervised by a USF faculty member and a corporate sponsor.

- CS 698 Directed Reading and Research (1-4 units)

  Written permission of the instructor, graduate program coordinator and dean is required.

- CS 699 Master's Thesis (4 units)

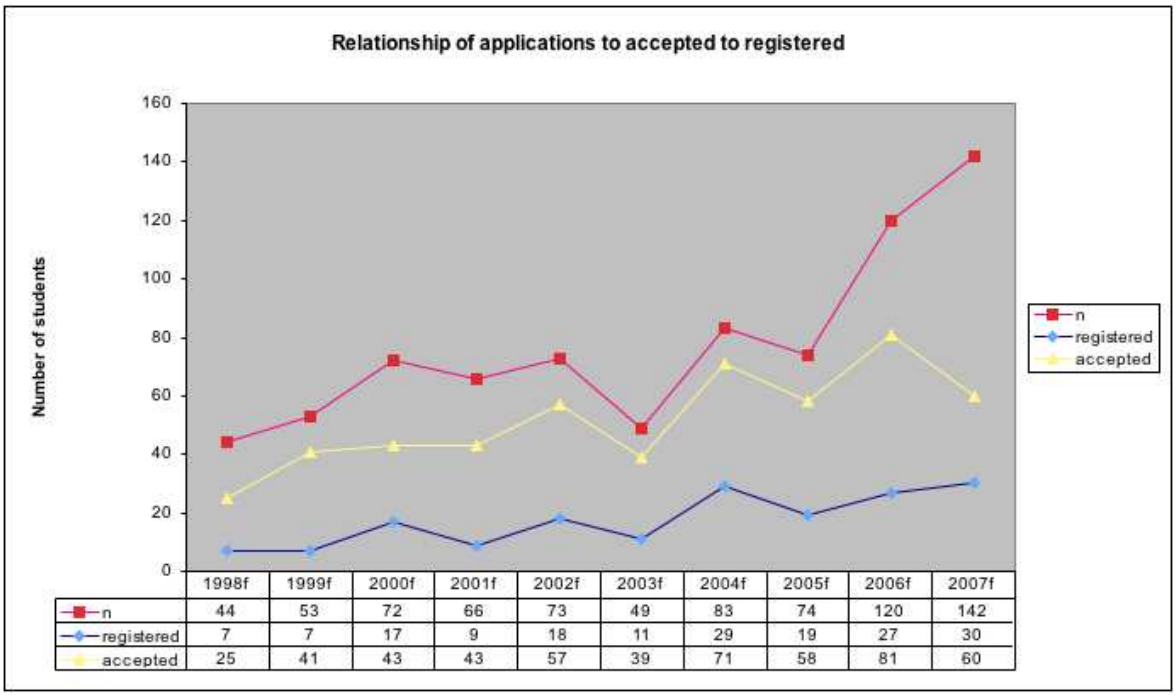
  Prerequisite: Thesis approval form required. Master's thesis.

4.4 Admission

Admission to both the MSCS and MSIE programs is based on the following information:

- GRE general scores
- GRE subject scores (optional)
- Official undergraduate transcripts
- 2 Letters of recommendation
- Applicant's background and experience essay

The admissions committee may contact recommenders, interview applicants, and request additional information to more accurately evaluate candidates for admission and fellowship awards. In addition, applicants must demonstrate a facility with programming via work experience, self-study, or coursework from a university providing practical coding experience as well as theoretical knowledge. Currently, our acceptance rate is approximately 52% (accepted applications / complete applications). The percentage of students that accept our offer (the yield) is 50%. The following graph provides historical data on the number of applicants, the number of accepted students, and the number who register.



	1998f	1999f	2000f	2001f	2002f	2003f	2004f	2005f	2006f	2007f
n	44	53	72	66	73	49	83	74	120	142
registered	7	7	17	9	18	11	29	19	27	30
accepted	25	41	43	43	57	39	71	58	81	60

It is worth mentioning that the average GRE math scores for accepted graduate students is 758. In addition, the graduate program director has visited India (2005) and China (2007) to give lectures and recruit graduate students.

4.4.1 Required Course Background

For admission, all students must provide evidence of the following coursework.

- High-level Programming (CS 110: C, C++, Scheme, ML, Java, etc.)
- Object-oriented Programming (CS 112: C++, Java)
- Low-level Assembly Language/Systems Programming (CS 210: Intel preferred)
- Calculus and Analytic Geometry (Math 109: differential & integral; business calculus not accepted)
- Linear Algebra and Probability (Math 201)
- Discrete Math (Math 202)

The following courses are not required for admission to the graduate program, but must be completed (at USF or elsewhere) to achieve regular status as a graduate student.

- Algorithms and Data Structures (CS 245)
- Computer Architecture (CS 315)
- Operating Systems (CS 326)
- Compilers (CS 414 or both of the following: CS 345 Programming Language Paradigms and CS 411 Automata)

The Internet Engineering program is just as rigorous, but requires fewer background courses because it focuses on Internet application development rather than classical MSCS topics such as operating systems and automata theory:

- Introductory programming (two courses)
- Upper-division programming (one course)
- Discrete math (one course)
- Data structures (one course)

4.4.2 Computer Science Background Essay

All applicants must submit a "Computer Science Background Essay". This two- to three-page essay, double-spaced, is to be submitted to USF's Office of Graduate Admission along with the other application items. The department is interested in highly-motivated students with strong technical backgrounds who are eager to develop innovative software and to perform research, as well as extend their computer science education. This essay provides an opportunity for applicants to detail their software development, research and extracurricular experiences. The essay is used in conjunction with other quantitative measures to evaluate students' eligibility for admission and for a Graduate Fellowship. In the essay, we are interested in an applicant's computer science experience and background, including the topics listed below. Applicants must describe their role in these endeavors and what technology and languages they used:

- Most significant software development projects. Including the purpose of the project – commercial, academic, or just for fun.
- Research projects
- Published articles or other technical documents

- Significant skills and interests beyond computer science, especially those that show creativity
- Experience communicating technical material such as teaching or making presentations
- Management or team leadership roles
- Experience in public service projects

4.5 Fellowships

A number of research fellowships are granted each year. The intention is that recipients of fellowships will engage in research under the direction of faculty members. These fellowships are awarded to the top applicants purely on the basis of merit. To continue receiving units, awardees must earn a GPA of 3.3 or better and be involved in a research project with a faculty member after their first semester. Awards are reviewed each semester. *Currently students are not actually required to do research with faculty.* No mechanism exists to decide which students go to which faculty and no procedures are in place to track student research. We have not discussed how much research they would have to perform in exchange for how much fellowship money.

4.6 Advising

All graduate student advising is done by the graduate director. Because of the large number of students (currently about 60), an *en masse* registration approach is taken. Students are advised as a group and then asked to register online. Students with special problems are encouraged to see the graduate director one on one. The graduate director tracks the progress of graduate students by forcing students to get an advising "hold" removed from their account before they can register online. The graduate director helps students choose the most appropriate set of courses based on their background and remaining requirements. Students in tenuous positions are advised away from directed studies, the practicum option, and any other activity that jeopardizes their continued presence at the University of San Francisco. As many as one or two students a year are dismissed. At any one time there may be as much as 15% of the student body on probation.

4.7 Academic Quality

Graduate students must maintain a 3.0 GPA to be considered in good standing. Students that drop below 3.0 are placed on academic probation and have one semester in which to bring up their overall average.

Except in unusual circumstances, students on probation for more than one semester are dismissed from the University. Students that have been promised Fellowship money must maintain a 3.3 GPA or lose the money immediately after a semester in which they drop below 3.3. Courses are 4 units and, hence, 2 courses are considered full-time. Faculty expect up to 20 hours per week of outside work for each class, implying a 40 hour week plus course time of 4 * 1:45 = 7 hours.

Of key importance to the continued success of our graduate program is a consistent level of rigor across all graduate courses. The faculty ask that the external reviewers evaluate the courses taught on a regular basis for rigor and educational content.

5 Faculty

5.1 Gregory D. Benson

Gregory D. Benson is Associate Professor of Computer Science, and Department Chair of Computer Science. He received his Ph.D. from the University of California, Davis, in 1999. He joined the Computer Science Department as an Assistant Professor in 1998. His research areas include operating systems, parallel computing, and programming languages. He is currently working on distributed systems and virtual machines. His teaching areas include operating systems and computer architecture.

5.2 Christopher H. Brooks

Christopher H. Brooks is Assistant Professor of Computer Science. He received his Ph.D. from the University of Michigan in 2002. He joined the Computer Science Department as an Assistant Professor in 2002. His areas of interest include artificial intelligence, multiagent systems, social aspects of computing, information economics, and machine learning. His teaching areas include artificial intelligence, distributed systems, and introduction to computer science. He is also the director of the department's Community Connections service learning program.

5.3 Jeff T. Buckwalter

Jeff T. Buckwalter is Associate Professor of Computer Science. He received his Ph.D. from Carnegie-Mellon University in 1978. He joined the Computer Science Department as an Assistant Professor in 1982. His areas of interest include performance of parallel computer systems, computer networks, operations research, and computer applications. His teaching areas include computer networks, project-based courses, and practical computer application courses.

5.4 Allan Cruse

Allan Cruse is Professor of Mathematics and Computer Science. He received his M.A. from the University of California, Berkeley in 1965, and his Ph.D. from Emory University in 1974. He joined USF in 1966. His current research interests are in the areas of systems programming for micromputers and combinatorial optimization. He teaches courses in advanced microcomputer and advanced systems programming, and also teaches in the Mathematics Department.

5.5 James K. Finch

James K. Finch is Professor of Mathematics and Computer Science. He received his Ph.D. from the University of Illinois in 1972. He joined USF in 1972. His areas of interest include computational statistics and scientific visualization. He teaches in the Mathematics Department, and also computer graphics and numerical analysis.

5.6 David Galles

David Galles is Assistant Professor of Computer Science. He received his Ph.D. from the University of California at Los Angeles in 1997. He joined the Computer Science Department as an Assistant Professor in 1997. His areas of interest include causal networks, programming languages, and artificial intelligence. He teaches data structures and algorithms, automata theory, and compilers.

5.7 Peter S. Pacheco

Peter S. Pacheco is Professor of Mathematics and Computer Science. He received his Ph.D. from Florida State University in 1983. He joined USF in 1989. His areas of interest include parallel computing, algorithms, and numerical analysis. He teaches parallel computing, data structures and algorithms, and courses in the Mathematics Department.

5.8 Terence Parr

Terence Parr is Assistant Professor of Computer Science. He received his Ph.D. from Purdue University in 1993. He joined the Computer Science Department as an Assistant Professor in 2003. His interests include his ANTLR parser generator, the jGuru developers web site, and his StringTemplate engine. His areas of teaching include software development and programming languages.

5.9 Sami Rollins

Sami Rollins is Assistant Professor of Computer Science. She received her Ph.D. from the University of California, Santa Barbara in 2003. She is also an Adjunct Professor at the University of Massachusetts, Amherst. She joined the Computer Science Department as an Assistant Professor in 2006. Her research interests include energy management for mobile devices. Her teaching areas include Internet systems research, project-based courses, and introduction to computer science.

5.10 Kim D. Summerhays

Kim D. Summerhays is Professor of Chemistry and Computer Science, and Chair of the Chemistry Department. He received his Ph.D. from the University of California, Davis in 1971. He joined USF in 1973. His research focuses on aspects of computer graphics and artificial intelligence. He teaches chemistry courses, and project-based courses in computer science.

5.11 Benjamin Wells

Benjamin Wells is Professor of Mathematics and Computer Science. He received his Ph.D. from the University of California, Berkeley in 1982. He joined USF in 1983. In his research areas he works on the boundary of logic, algebra, and computing; and also contributes to computer graphics, visual communication, and classic computers. His teaching areas include logic, computer graphics, freshman seminars that combine science and art, and also mathematics courses.

5.12 David Wolber

David Wolber is Professor of Computer Science. He received his Ph.D. from the University of California, Davis in 1991. He joined the Computer Science Department as an Assistant Professor in 1993. His areas of interest include programming by example, intelligent user interfaces, internet systems and development, and digital libraries. His courses include internet programming, projects-based courses, and introduction to computer science.

6 Staff

6.1 Alex Fedosov

Alex Fedosov is Manager of Scientific Computing for the Computer Science Department. He manages the four supercomputers housed at the Computer Science Department, as well as oversees the Department's server and network infrastructure. He received his B.S. in Computer Science from the University of San Francisco. His research interests include systems programming and parallel and distributed systems.

6.2 Cody Nivens

Cody Nivens is the System Administrator for the College of Arts and Sciences, and provides technical support for the Computer Science Department, manages computer resources for the Arts & Sciences and implements special projects for various departments. He received his B.S. in Computer Science from Cal Poly, San Luis Obispo and his M.A.O.M. from Santa Barbara College of Oriental Medicine.

6.3 Rosa Maria Garay

Rosa Maria Garay is the Program Assistant for the Computer Science Department.

7 Students

7.1 Undergraduate Students

Most of our undergraduates are from the United States. Many come from the San Francisco Bay Area. As explained in Section 3, our undergraduates come in with a wide range of preparation for computer science. Many have never taken a computer programming class or written a program at all. However, each year, one or two students come in with quite a bit of programming experience and perhaps good scores on the Computer Science AP exam.

Most of our students are not strong in mathematics and they are not enthusiastic about their required Math and Physics courses. We do not require much mathematics in the undergraduate major courses, so the Math courses tend to be terminal rather than serve as prerequisites for advanced CS courses. Also, as pointed out in Section 3, most of our attrition occurs in the sophomore year when the CS classes tend to become a bit more difficult. Also, the sophomore year is a good time to switch majors at USF, because many of them can be completed in three years.

An open question is how to recruit undergraduates from the existing USF student population. USF's enrollments as a whole are reaching capacity, and many other departments must turn away students. The opportunity exists for us to attract strong undergraduates, but outreach and publicity is an ongoing challenge.

7.2 Graduate Students

A recent strength of our program has been the growth of our graduate program. We have been able to compensate for the nationwide decline in undergraduate Computer Science majors through aggressive growth at the Master's level, particularly from international students. This has infused the department with new energy and helped to increase the amount of faculty research being conducted.

A downside to our increase in the number of international students is that many of them do not have the same software development and programming experience that we find with U.S. students. We often require deficiency courses for our graduate students who clearly do not have the core undergraduate CS background. This can be challenging when it appears a student has taken a core course in their home country, but the course lacked the content and focus in our equivalent courses. Many professors find that they must teach quite a bit of remedial material in the beginning of each graduate course to compensate for a lack of preparation. We are interested in finding ways to better vet international candidates before their arrival at USF, and to bring them up to speed more quickly.

8 Diversity

8.1 The Value of Diversity

Diversity is often presented as a goal to be worked toward without a real discussion of *why* we as a department or university should be interested in diversity. Sadly, it often gets placed in an either-or setting with quality, with the implication being that a department can either choose to have the very best, or it can choose to be diverse. This is a false dichotomy. In fact, diversity can be synonymous with excellence. By creating an environment where all students (or faculty) feel welcome and engaged, we can attract the best students to our major, rather than just male students, or Asian students, or white students.

The obvious reason to pursue diversity is the argument from social justice. Fairness and equality dictate that all races, sexes, and economic classes should have equal opportunity and representation. While this is true, and certainly directly in line with USF's mission, it is hardly the only reason to think about diversity. As educators, it is important to have a diverse faculty population because we have a diverse student population. If we are to best understand how to teach our students and to provide role models and examples for them, it is important for the students to have faculty whom they can identify with.

8.2 Student Diversity

USF as a whole is one of the most diverse universities in the US. US News and World Report's most recent college ranking placed USF in the top 20 in terms of student ethnic diversity and international student enrollment. Our Fall 2006 freshman class was 39% White, 21% Asian-American, 14% Latino, 6% multiethnic, 5% international, 4% African-American, 2% Pacific Islander, and 1% Native American (with 8% unidentified).

At the undergraduate level, our department's ethnic makeup is 33% White, 25% Asian-American, 28% International, 7% Latino, 2% African-American, and 2% multi-racial. At the graduate level, our student body is approximately 70% international (primarily Chinese and Indian) with the remainder being 14% Caucasian, 5% Asian-American, 1.7% Latino, 1.7% Pacific Islander, and 7% unspecified.

Our student body as a whole is approximately 60% female. Within the College of Arts & Sciences, the breakdown is 66% female and 34% male. Within our department, at the undergraduate level we have 42 males and 5 females, or 89% males. At the graduate level, we have 16 females and 42 males, or 73% males.

If we look at the national data as collected in the Computing Research Association's Taulbee Survey

from 2006 [2], we see that nationally, recipients of a B.S. in Computer Science are 85% male, with 63% White, 17% Asian, 8.7% international, 3.9% African-American, and 4.6% Latino.

We are relatively happy with our ethnic diversity, although we (and the university as a whole) could do a better job of targeting African-American students. At the graduate level, our primary recruiting focus has been international, and so that has had a profound effect on our ethnic makeup.

In terms of gender, it seems clear that we need to do a better job of targeting female students, especially at the undergraduate level, and particularly given the gender imbalance toward women within the University as a whole. Determining how to increase the enrollment of female students is an ongoing question. While there have been a number of strategies around better retaining women in science and engineering programs, it is less clear how to attract new students.

Here at USF, we have implemented several strategies to better target and nurture female students, including participation in USF's Women in Science organization, the construction of a Women in Computer Science group (led by Sami Rollins, our first tenure-track female professor), and offering a Summer Enrichment Program to expose young women (ages 13-18) to Computer Science. It is unclear what effect these programs have had in recruiting new female students, but they have certainly helped develop a more welcoming, community-oriented atmosphere for our existing women students.

8.3 Faculty Diversity

It seems clear that diversity in the student body and diversity among the faculty are connected; as was mentioned above, it's important for students to have role models that they can relate to and identify with. While this does not have to occur purely along ethnic or gender boundaries, it would be naive to think that this is not important. Until 2006, our department was composed entirely of Caucasian male faculty in the tenure-track category, although there had been several unsuccessful attempts to hire female and/or ethnic minority candidates. In Fall 2006, Sami Rollins joined the department as a term position and, in Fall 2007, she became our first female tenure-track professor.

This is an area that definitely needs attention. If we want to attract more female students, it is important for them to have mentors and for us to foster a community of inclusion. If we are to continue our aggressive recruiting of international students, it is important to have faculty who these students can identify with and relate to. Faculty with an international background can also help us recruit more effectively and better vet the applications of foreign students.

[2] http://www.cra.org/statistics/

9 Assessment

9.1 Learning Outcomes

Here are the stated learning outcomes for the CS undergraduate major. Students who complete the B.S. in Computer Science will be able to demonstrate:

- An understanding of fundamental topics in computer science including programming, data structures, algorithms, and computer systems implementation;
- The ability to design, implement, and debug software applications;
- Effective communication and team participation skills with respect to software development.

Students who complete the Masters of Science in Computer Science will be able to demonstrate:

- An understanding of advanced topics in computer science including software engineering, algorithms, artificial intelligence, programming languages, parallel computing, networking, and low-level systems;
- The ability to design, implement, and debug large-scale software applications;
- The ability to evaluate and understand advanced research from computer science literature;
- Effective communication and team participation skills with respect to software development.

We currently do not have a formal means to assess if our students are achieving these learning outcomes. Anecdotal evidence suggests that we are doing a good job with respect to these outcomes. An open question is how to introduce formal assessment in a manageable fashion.

9.2 Specific Forms of Assessment

We assess student performance in all of the traditional ways:

- Homework assignments,
- Midterm examinations,
- Programming assignments, and
- Final examinations.

Some of our faculty also use alternative methods of assessing student performance.

- Professor Benson uses "interactive" grading for the programming assignments in his operating systems class. Students sit down at a computer with Professor Benson and answer questions about their programs. In addition to making sure that the students actually understand their programs, this approach has a couple of additional benefits. Students who work on group projects aren't assessed simply on the basis of the finished project: a student's contribution to the project can be assessed on the basis of his or her understanding of the solution. Also students who plagiarize a program or part of a program are invariably unable to explain how the program works, and, as a consequence, are immediately apprehended.

- Professor Cruse exploits the the interactive capabilities of the Kudlick classroom and the 2-hour duration of the class-period by including an "online+open-book" programming exercise as a component of his examinations. There are three such exams during the semester, each one having two parts: Part One is a traditional closed-book university-style exam with a mix of question-types (e.g, concept definitions, short-essays, computations, code-fragment analyses, algorithm-design, et cetera) accounting for roughly 80% of the exam score. This part usually consumes about half of the available exam-time. Part Two is an online programming question, but any books or class-notes, as well as information available on the Internet may be consulted, although communication with anyone (other than the the instructor) is not permitted, and students submit their programming solution electronically for testing by the instructor. They also submit a printed copy on which the instructor can write comments.

  Although students typically seem intimidated at first by the prospect of a time-constrained programming design problem, their level of apprehension tends to diminish as the course progresses, and indeed their level of programming self-confidence generally rises as they discover what they are capable of doing even under some time-pressure.

- Professor Wells follows the pattern of Brian Harvey (Computer Science, UC Berkeley) and allows his students to retake some or all of an exam in a small group, with the group score contributing 25% or so to the individual scores of the group's members. This happens directly after the individual exam. It typically produces a positive reaction and eager discussion. Students leave the classroom with a clearer understanding of the exam and know what to expect when they see their own scores later. Often, but not always, Prof. Wells protects individuals against lower group scores.

- In the Special Lecture Series in Computer Science, Professor Wells' students are assessed in three ways:

  - Each students must write a report on the speaker's talk in the form of a news/feature story. This should consist of three to five sentences in correct English, with no editorializing.
  - Each student must also prepare a list of descriptors on the speaker's style,
  - Finally each student must discuss the presentation styles (not content) of three to five speakers at at time.

- In many of Prof. Wells' courses, students prepare and present project reports. Written and oral presentations are required. Some have involved the student giving other students homework.

We also assess faculty performance using a standardized form (SUMMA) given to the students in every class. Many faculty also administer their own surveys to their classes. For example, Prof. Pacheco uses the following basic template for each of his classes:

> What did you like or dislike about each of the following? Why? What would you change? Be specific. Continue on the back if you need more space.
>
> 1. The lectures/discussions.
> 2. The coursework: homework, programming assignments, exams.
> 3. The texts.
> 4. Other.

Although the results have not yet been made available, we understand that the University is also conducting a survey to assess the satisfaction of USF computer science majors with their education.

10 Governance

Governance the Department of Computer Science is relatively straightforward. Most governance is centered around the department chair. A department chair is elected for a three year term. The chair has the following responsibilities:

- Run monthly faculty meetings.
- Prepare the schedule of courses for each semester.
- Organize Fall and Spring orientation for new students.
- Works with the administration on initiatives.
- Handle student issues and complaints.
- Works with the program assistant to schedule advising, audit student progress.
- Organize the annual CS Night event.
- Work on curriculum changes.
- Manage the CS budgets and plan for infrastructure improvements.
- Serve as the main advocate for CS in campus-wide councils and forums.
- Help the administration maintain relationships with CS alumni.

Most curriculum changes are brought to the entire faculty for discussion. Changes are usually reached by consensus.

11 Technology

The computing resources provided and managed by the Department of Computing Resources are quite impressive given our size. We have a sophisticated *studio classroom* called the "Kudlick Classroom", two undergraduate labs, one graduate research lab, and three clusters for parallel computing and system experimentation. We have a minimum 3 year replacement policy for all of our equipment.

11.1 The Kudlick Classroom

The Kudlick Classroom supports an interactive hands-on teaching style with 30 student workstations, 2 laser printers, 2 instructor workstations, 2 plasma displays, camera and projector, videotape and DVD players, USB ports supporting audio and robotics, plus laptop, network, and WiFi connectivity. The classroom has transformed the way many of our classes are taught; now many classes use a combined lecture/laboratory format. The Kudlick classroom also serves as an additional student-use computer lab on evenings and weekends via One Card access.

11.2 CS Student Labs

The "PC Lab" (Harney 535) contains 12 to 16 dual-boot workstations (Linux and Windows XP), 1 laser printer, null-modem cables connecting adjacent stations (to support experiments programming character device-drivers). The lab is open 24 hours/day via One Card access.

The "Mac Lab" (Harney 530) contains 12 to 16 dual-boot (Linux and Mac OS X) Apple dual-processor workstations, with overhead projector supporting lecture-style presentations. The lab is open 24 hours/day via One Card access.

The "Graduate Research Lab" (Harney 536) provides additional lab space for CS graduate students and individual research projects directed by CS faculty. This lab has 4 PC workstations and 3 PowerMac workstations.

11.3 Cluster Computing

The "Penguin Cluster" consists of 24 dual-dual-core Opteron nodes connected by an Infiniband high-speed network and is used by parallel computing classes as well as for research.

The "Bass Cluster" consists of 24 dual-dual-core Opteron nodes connected by a Myrinet high-speed network and is used for teaching and research.

The "Anchor Cluster" consists of 16 Intel dual-core server stations with remote boot capability and private gigabit-speed LAN interconnection (in addition to departmental network access), supporting special topics instruction in Linux systems programming.

12 Plans for the Future

12.1 Goals

Our high-level goals for the future include:

- Continue to improve the curriculum and rigor in our programs to ensure we are preparing our students for computing careers and graduate school.
- Find innovative ways to attract more students to the CS major and retain those students.
- Seek out opportunities for interdisciplinary degree programs.
- Continue to maintain excellent computing resources for our students.

The remaining sections provides some detailed ideas to achieve these goals in our undergraduate program, our graduate programs, and with respect to our computing resources and teaching facilities.

12.2 Undergraduate Program

As pointed out in Section 1, like many CS departments across the nation, we are experiencing low enrollments in the Computer Science major. The strong enrollments in our graduate programs help make up for the low undergraduate numbers. Nonetheless, we would certainly like to improve our enrollment numbers. Much discussion in the department has focused on how to achieve better enrollments.

A major issue is the number of units required for the CS major. Should we reduce the total number of units? While we recognize many good reasons for reducing the unit requirement, we struggle with the fact that the current program as is does an excellent job at preparing students for their future as computer scientists. One reason it is so difficult to remove courses from the major is that there is a belief that we barely cover the minimum background that every CS major should have. The switch to a 4 unit course model required us to offer fewer "fatter" classes. As such, under the 3 unit model we were able to require a broader range of topics.

A challenge with the current model of teaching Computer Science, not only at USF but at most universities, is the "onion model" of structuring the curriculum, in which topics are introduced in successive levels of difficulty and abstraction, with each new course "peeling back the onion." This has some well-understood advantages pedagogically, but two potential weaknesses.

- It makes it very difficult to introduce new or cutting-edge topics such as gaming, pervasive or wearable computing, social networks, or Web-based computing into the curriculum. Ideally, it would be possible to infuse new topics throughout the undergraduate curriculum, but this requires a serious rethinking of the order and structure of topics in the major.

- The "onion model" may be a factor in student attrition, as students must spend several semesters learning fundamentals before they really engage with computing topics that reflect their interests, or their prior experience with computing. Providing students with early exposure to topics that more closely reflect their experience could serve to make the major more attractive.

Recognizing these weaknesses is one thing, but determining how to address them is another. Restructuring the curriculum would undoubtedly be a major undertaking, and one that is not to be entered into lightly.

In the Fall of 2007, the CS department held a one day retreat to discuss a range of issues facing the department. In terms of the undergraduate program, we came up with the following proposals:

- Develop a BA in Computer Science that requires fewer units.

- Change the two semester Physics requirement to two semesters of any lab science such as Biology or Chemistry. In addition, a student could choose to take more mathematics instead of the lab science requirement.

- Develop a Freshman Seminar Course to attract students.

- Better publicize faculty research.

- Introduce a "mentorship" program in which upper division students working on faculty research take on lower division students as assistants.

- Improve our web presence.

- Conduct entry and exit surveys.

12.3 Graduate Programs

An important goal of our department is to increase the size of our graduate program while at the same time increasing selectivity. Naturally, this requires a large increase in the number of applicants. The graduate program director is working closely with the faculty and administration to bring in large numbers of new applicants. One way in which we hope to draw in students is to offer generous fellowships to students that

would not normally choose USF. Because these students would be drawn from a new market, the number of students should increase as will the quality. A larger student body makes additional faculty lines possible and higher-quality students provide excellent research assistants as well as a more pleasurable teaching environment. As of Fall 2007, the administration has not approved an increase in fellowship money to draw in such high-quality students nor has the amount of fellowship money kept pace with increases in student body size.

Faculty research is an important "marketing" factor that makes the University more attractive to prospective students. Hiring more faculty and promoting research among the existing faculty will draw in more and higher-quality students. One way in which we can increase research within the department is to regularly share research ideas, plans, and current status with each other. The faculty should get together once a semester and have a research roundtable where each faculty member could discuss their research.

Increased intra-department communication could also help us become better teachers as we learn strategies for success from our fellow faculty. During the roundtable meeting, faculty could also briefly describe their syllabi and course designs.

At a retreat in the fall of 2007, the faculty agreed that the term "Internet Engineering" was misleading as it evokes an image of routers and ethernet cables. A more appropriate term is probably "Web Science". For example, webscience.org is a joint research initiative between MIT and University of Southampton. The initiative's goal is to "... examine the World Wide Web and offer the practical solutions needed to help guide its future use and design." Also related to our Internet Engineering program, the faculty would like to revisit two courses. An existing course called *Human Computer Interaction* should be dropped in favor of a new course. The focus of *Internet Systems Research* should also be changed because experience shows students have great difficulty reading research papers and writing discussions. To date we have not established the new name for fear that "Web Science" may not be any more understandable than "Internet Engineering."

12.4 Facilities and Technology

The CS department is proud to provide excellent computing resources and teaching facilities to our students. We would like to continue to maintain and improve our resources. Here is an overview of the evolution of the CS department over the last 10 years:

- Growth from one lab to five: PC lab, Mac Lab, the Kudlick classroom, and two research labs
- Transition from standard Dell machines to sophisticated cutting-edge hardware

- OS switch from DOS and single-user Windows only to multi-user Windows XP, Linux, Mac OS X
- A switch from no user accounts to an LDAP database and networked home directories
- Growth from three AIX servers to a sophisticated server infrastructure (mail, web, NFS, LDAP, DHCP, DNS, etc.)
- The addition of three clusters and custom setups for classes

The Support staff growth did not keep pace with our technology growth. We had one full-time staff person taking care of everything 10 years ago and currently, we only have 1.5 full-time staff.

Our recommendation in order to continue providing quality system administration service to the department is to have 3 full-time staff members, each dedicated to various duties:

- One dedicated full-time to labs, classroom, and all things related (lab images, maintenance, A/V, etc.)
- One to maintain and improve/upgrade/replace the servers and network infrastructure (Internet access, mail, web, NFS/SMB, backup, etc.)
- One to concentrate on the clusters and special projects and to provide programming support.