2013 Self-Study Report:
Computer Science Department's Graduate Programs

Master of Science in Computer Science (initiated in 1975, last program review 2005)

Master of Science in Software Engineering (initiated in 2007)

Graduate Coordinator: Ani Nahapetian

Previous Graduate Coordinator: Robert Lingard

Department Chair: Steven Stepanek

Signatures:

Department Chair: _____ Date: _____

College Dean: _____ Date: _____

G. Date report completed and submitted to Undergraduate Studies

VERIFICATION OF PROGRAM FACULTY REVIEW

The Computer Science Department faculty were involved in the preparation of the self-study report. Excerpts from relevant department meeting minutes where the self-study report content was discussed are included in the appendix. Also, the full faculty was given an opportunity to review and edit drafts and sub-drafts of the document during the Fall 2012 and Spring 2013 semesters.

TABLE OF CONTENTS

SELF-STUDY

Overview of Master's Programs

The CSUN Computer Science Department offers two Master of Science (MS) graduate degrees, the Master of Science in Computer Science which has been offered since 1975 and the Master of Science in Software Engineering which was started in 2007. The aim of the MS programs is to provide a core of advanced courses which build upon the knowledge gained from an undergraduate degree in Computer Science. The core is complemented by a set of electives in advanced Computer Science and Software Engineering topics and project work for a thesis.

The Computer Science Department offers a wide range of classes and provides opportunities for thesis work in many different areas. Classes are offered in diverse areas including security, networking, embedded systems, e-commerce, graphical user interfaces, object oriented software development, artificial intelligence, architecture, and theory.

The Master of Science in Computer Science program is available to students with an undergraduate degree in Computer Science, as well as to students with other undergraduate majors who complete appropriate prerequisite courses. Students take a core of graduate level Computer Science classes and select from a wide range of advanced elective courses. Possible career paths with the degree include software development, systems analysis, systems design, networking, security and database design. The degree is also a good stepping stone to more advanced degrees, such as a Ph.D. in Computer Science.

The Master of Science in Software Engineering degree is open to students with undergraduate degrees in Computer Science, Software Engineering, and Computer Information Technology, as well as to students who complete the prerequisite courses. The objectives of the Software Engineering program are to develop students' expertise in the area; to enable students to learn emerging concepts in the field; to meet the software engineering needs of the working professional; to meet the software engineering needs of technological organizations and industry; and to prepare students for subsequent graduate work in Software Engineering. Potential employment opportunities that are open to graduates include software project management, software system design and development, and software quality assurance.

In each program students must complete 30 units of graduate work, including 6 units involving a thesis or graduate project. The core of the M.S. Program in Computer Science comprises advanced courses in computation theory, algorithms and data structures, system architecture, and software engineering. For the M.S. Program in Software Engineering advanced courses in software engineering processes, including requirements analysis, software design and implementation, verification and validation, quality assurance, software maintenance, and software project management make up the core. The electives in either program may be chosen

to form a concentration in an area of specialization or to provide a broadly based program of study, whichever is more consistent with the selected thesis or graduate project. The course objectives of all courses which are a part of the graduate programs can be found in Appendix A. For each course the specific course objectives are also given.

All courses in the student's graduate program must be completed with a grade of C or better. No course taken more than 7 years prior to the date of which all requirements for the degree are completed may be counted as part of the 30 units in the degree program. No time limit applies to courses taken to satisfy prerequisite requirements.

For admission to either program, applicants must meet not only the requirements of the University as listed in the Catalog, but they must have a 3.0 GPA and score in the 50th percentile on each of the three parts of the Graduate Record Examination (GRE). Applicants who have completed an ABET-accredited Computer Science or Software Engineering Bachelor of Science program and have meet all other entry requirements are exempt from the GRE requirement. Additionally, students must complete prerequisites as identified for each program before attaining full classification under the program.

Each M.S. candidate must submit a proposal for a thesis or graduate project to be done under the supervision of a faculty member. When the thesis or project is approved by that faculty member, the graduate coordinator and the Department Chair, the proposal becomes a contract between the student and the Department as to the work to be done for the thesis or graduate project. A three member graduate committee is formed with that faculty member as its chair. When the work is done, the student must prepare a thesis and defend the results of the thesis or graduate project before the committee. The thesis and presentation must be approved by the student's committee.

Course Requirements for the Master of Science in Computer Science

Students must complete one course from three of the following four areas (courses available for selection in each area are shown):

Algorithms:

- COMP 610 Data Structures and Algorithms (3)

Systems:

- COMP 620 Computer System Architecture (3)

Software Engineering:

- COMP 680 Software Engineering (3)

Foundations:

- COMP 615 Advanced Topics in Computation Theory (3)
- COMP 630 Formal Semantics of Programming Languages (3)

Additionally students must complete the following two Thesis/Project courses:

- COMP 696 Directed Graduate Research (3)
- COMP 698 Thesis or Graduate Project (3)

Finally, students must complete 15 units of electives selected from 400, 500, or 600 level Computer Science courses (not including COMP 450, 480/L, 490/L, 491L, 494, 499, 696, 698,

and 699). The student's Thesis/Project Committee Chair may require that specific elective courses be taken prior to enrollment in COMP 696 and COMP 698. Students should seek approval from their Committee Chair prior to enrolling in elective courses. At least 6 units must be at the 500 level or above.

Course Requirements for the Master of Science in Software Engineering

The following is a list of core courses that each student in the programs must successfully complete:

- COMP 682 Requirements Analysis and Specification (3)
- COMP 684 Software Architecture and Design (3)
- COMP 680 Advanced Topics in Software Engineering (3)
- COMP 686 Software Engineering Management (3)

Additionally students must complete the following two Thesis/Project courses:

- COMP 696 Directed Graduate Research (3)
- COMP 698 Thesis or Graduate Project (3)

Unlike the Computer Science master's program, in Software Engineering group projects are allowed. In the case of a group project, each member of the group must present and defend his/her contribution to the final result. The accompanying report must clearly identify the contributions of each member of the group. Each member of the group is evaluated separately by the committee. The report and presentation or relevant portion for each member of a group must be approved by the Project/Thesis Committee.

Finally, students must complete 12 units of electives selected from 400, 500, or 600 level Computer Science courses (not including COMP 450, 480/L, 490/L, 491L, 494, 499, 696, 698, and 699). Two of the elective course must be from the following list:

- COMP 541 Data Mining (3)
- COMP 581 Open Source Software Engineering (3)
- COMP 584 Advanced Web Engineering (3)
- COMP 585 Graphical User Interfaces (3)
- COMP 586 Object-Oriented Software Development (3)
- COMP 587 Software Verification and Validation (3)
- COMP 589 Software Engineering Metrics (3)

The student's Thesis/Project Committee Chair may require that specific elective courses be taken prior to enrollment in COMP 696 and COMP 698. Students should seek approval from their Committee Chair prior to enrolling in elective courses. At least 6 units must be at the 500 level or above.

Steps to Completing Thesis Work

The following steps are necessary to completing the thesis requirement in both M.S. programs.

1. Student obtains classified status.
2. Student selects a Computer Science Department faculty member who will agree to serve as the graduate advisor/committee chair. Students choose an advisor by speaking

with members of the faculty whose research interests best match theirs. The advisor may request that students take certain classes, complete a specific project, and/or follow certain guidelines when completing the thesis work.

3. Students often form a committee after having completed the core classes and having begun taking some electives. This gives them an the opportunity to choose some of the electives so they are relevant to the thesis topic.
4. A typically 5 pages long proposal is prepared and approval is obtained from the committee chair and the graduate coordinator.
5. With the help of the committee chair, two other committee members are chosen. Committees must be composed of at least three people, including the committee chair. Aside from the committee chair, another Computer Science faculty member must serve as a committee member. The third member may be from any department, including Computer Science. Normally this member is from the College of Engineering and Computer Science, but may be from any department at CSUN or from off-campus. If an off-campus individual is selected to be the third committee member, the graduate coordinator's approval is obtained. Students discuss with each committee member what they expect in their thesis and the procedures to follow in seeking their approval. All committee members, including the committee chair, must agree to serve on the committee by approving the planning form on the Electronic Thesis and Dissertation (ETD) system. Also, the graduate coordinator must approve the committee on ETD.
6. The student is then ready to enroll in and successfully complete COMP 696C. To enroll in COMP 696C, an R-form must be completed and approved. To receive credit for COMP 696C, a proposal must have been approved by the committee and the graduate coordinator, a committee must have been formed, and all the project work must be complete.
7. In a following semester the student enrolls in and successfully complete COMP 698C. To enroll in COMP 698C, an R-form must also be completed and approved. Students prepare and submit their thesis on ETD during this time. Thesis drafts are reviewed by the committee chair numerous times throughout this final semester.
8. A Master’s defense is carried out by the student. To prepare for the defense, students must visit two Master's defenses before presenting their own defense.

Enrollment Trends

Tables 1, 2, and 3 show the enrollment trends over the last seven years by major. The first shows data for Computer Science majors, the second for Software Engineering majors, and the third gives the combined results. For each semester, the tables show the number of applications received, the number that were admitted, the acceptance rate, number of admitted students who actually enrolled, and the yield rate for accepted students (the percentage of accepted students who actually enrolled). Tables also show for that semester the total number of graduate students taking classes, the number of those graduating, and, for those graduating, the average number of semesters it took to obtain the degree.

The data show a slight decline in the number of Computer Science students over the last seven years which has been offset by an increase in the number of those in Software Engineering. For the most part the combined totals show little variation over the seven year period although it appears that the total enrollment numbers have been steadily growing since 2012. In fact, the number of graduate students enrolling in 2012 was more than 40% higher than the number who enrolled in 2010.

Looking at the combined totals, it seems that the department receives about 300 applications each year and nearly 60 of those are accepted. The actual average acceptance rate is just over 17%. Of those about 37% actually enroll. That means we enroll about 21 new students each year. On the average nearly 13 students successfully complete the master's program each year. Assuming we are in a steady state position that means our graduation rate is just over 60%. In an attempt to validate that number using another method, we looked at all students who first enrolled in 2006 and 2007. There was a total of 42 such students and 23 of those graduated. Of the remaining 19, 17 have left the program and two are still currently enrolled. Ignoring the two who are still in the program, it means the graduation rate for this group of students is 57.5%, close to the previous estimate.

As far as the time required to complete the degree is concerned, we see that the overall average is between six and seven semesters. This seemingly long time to graduation is likely due to the fact that many of our Master's students are working professionals pursuing an advanced degree on a part time basis. In other studies it has been shown that the average number of units taken by a graduate student each semester is only about six. Furthermore, many of our students enter the program without undergraduate degrees in a computer related major and must take several prerequisite courses as well. This adds to the time to earn the degree.

The stories for both the Computer Science and the Software Engineering graduate programs are similar in many respects. The biggest difference is in the number of applications received and the percentages admitted and enrolled. For Computer Science only 16% of applicants are accepted and only 36% of these enroll. For Software Engineering, the corresponding numbers are 26% and 40%. These differences are probably due to the fact that many international students apply to the Computer Science Program, but few choose Software Engineering. These international students seldom meet the rigorous entrance requirements for our programs and hence contribute disproportionately to the low acceptance rate. Those students applying for the

Software Engineering Program tend to be local students, many of whom have graduated from CSUN. These students have a much higher probability of being accepted, and if accepted, are more likely to enroll.

Based on the data, the increasing need for Computer Science and Software Engineering expertise, and the improving economic climate within the state and the CSU, we predict that both of these programs will begin to grow significantly. The Fall 2012 numbers of new graduates for both programs were 40%-50% higher than they were two years ago. We expect this growth to continue to accelerate.

Computer Science	#Apps. Rec'd	#Admits	%Admit	#Admits Enrolled	%Admits Enrolled	#Grad. Students Attending	#Students Graduated	Average Sems. to Graduate
S06	34	10	29.41%	8	80.00%	76	10	7.50
F06	110	35	31.82%	13	37.14%	78	4	8.00
S07	58	15	25.86%	9	60.00%	72	4	6.00
F07	195	37	18.97%	12	32.43%	57	5	7.00
S08	149	13	8.72%	6	46.15%	57	4	8.00
F08	238	32	13.45%	12	37.50%	57	4	5.75
S09	147	16	10.88%	6	37.50%	55	7	6.86
F09	210	32	15.24%	11	34.38%	52	5	6.20
S10	89	14	15.73%	3	21.43%	51	7	8.40
F10	173	30	17.34%	8	26.67%	49	4	4.75
S11	58	7	12.07%	4	57.14%	44	5	9.50
F11	231	35	15.15%	10	28.57%	43	9	5.86
S12	109	9	8.26%	2	22.22%	43	6	6.40
F12	212	37	17.45%	12	32.43%	47	4	6.25
Total	2013	322	16.00%	116	36.02%		78	
Average	143.79	23.00	16.00%	8.29	36.02%	55.79	5.57	6.83
Minimum	34.00	7.00	8.26%	2.00	21.43%	43	4	3
Maximum	238.00	37.00	31.82%	13.00	80.00%	78	10	20

Table 1: Computer Science Graduate Program Enrollment Data

Software Engineering	#Apps. Rec'd	#Admits	%Admit	#Admits Enrolled	%Admits Enrolled	#Grad Students Attending	#Students Graduated	Average Sems. to Graduate
S08	7	2	28.57%	2	100.00%	6	1	4.00
F08	38	6	15.79%	2	33.33%	10	0	
S09	21	1	4.76%	0	0.00%	7	0	
F09	33	15	45.45%	8	53.33%	20	0	
S10	20	4	20.00%	1	25.00%	21	0	
F10	38	13	34.21%	4	30.77%	26	0	
S11	9	1	11.11%	0	0.00%	23	3	5.67
F11	50	17	34.00%	4	23.53%	18	4	5.33
S12	24	1	4.17%	1	100.00%	15	1	
F12	50	15	30.00%	8	53.33%	25	1	11.00
Total	290	75	25.86%	30	40.00%		10	
Average	29.00	7.50	25.86%	3.00	40.00%	17.10	1.00	6.00
Minimum	7.00	1.00	4.17%	0.00	0.00%	6	0	3
Maximum	50.00	17.00	45.45%	8.00	100.00%	26	4	20

Table 2: Software Engineering Graduate Program Enrollment Data

Comp. Sci. & Soft. Eng.	#Apps. Rec'd	#Admits	%Admit	#Admits Enrolled	%Admits Enrolled	#Grad Students Attending	#Students Graduated	Average Sems. to Graduate
S06	34	10	29.41%	8	80.00%	76	10	7.50
F06	110	35	31.82%	13	37.14%	78	4	8.00
S07	58	15	25.86%	9	60.00%	72	4	6.00
F07	195	37	18.97%	12	32.43%	57	5	7.00
S08	156	15	9.62%	8	53.33%	63	5	7.00
F08	276	38	13.77%	14	36.84%	67	4	5.75
S09	168	17	10.12%	6	35.29%	62	7	6.86
F09	243	47	19.34%	19	40.43%	72	5	6.20
S10	109	18	16.51%	4	22.22%	72	7	8.40
F10	211	43	20.38%	12	27.91%	75	4	4.75
S11	67	8	11.94%	4	50.00%	67	8	7.86
F11	281	52	18.51%	14	26.92%	61	13	5.70
S12	133	10	7.52%	3	30.00%	58	7	6.40
F12	262	52	19.85%	20	38.46%	72	5	7.20
Total	2303	397	17.24%	146	36.78%		88	
Average	164.50	28.36	17.24%	10.43	36.78%	68.00	6.29	6.74
Minimum	34.00	8.00	7.52%	3.00	22.22%	57	4	3
Maximum	281.00	52.00	31.82%	20.00	80.00%	78	13	20

Table 3: Combined Computer Science & Software Engineering Graduate Program Enrollment Data

Instructional and Support Staff

The Computer Science Department has 18 tenure or tenure-track faculty members. Since the last program review, two new tenure-track faculty have joined the department. Two part-time faculty members have taught graduate courses in the department, since the last review. The Computer Science Department does not currently employ any teaching assistants in its programs.

Specifically, the following faculty have been involved in teaching courses in the Computer Science and Software Engineering graduate programs over the period of this review. Excluding the two part-time faculty, all sixteen are tenured or tenure-track members of the faculty. The CVs of these faculty members can be found in the Appendix.

Faculty Teaching Core Courses in the Graduate Programs in Computer Science and Software Engineering

Jack Alanen (FERP) – Software Engineering
Shan Barkataki – Computer Science and Software Engineering
Richard Covington – Computer Science
Patricia Dousette (part-time) – Computer Science and Software Engineering
Steven Fitzgerald – Computer Science
Robert Lingard (FERP) – Computer Science and Software Engineering
Richard Lorentz – Computer Science
Robert McIhenny – Computer Science
John Noga – Computer Science

Additional Faculty Teaching 500 Level Electives in the Graduate Programs

Michael Barnes
Peter Gabrovsky
Son Pham
Gloria Melara
Ani Nahapetian
Brenda Timmerman (retired/deceased)
George Wang
Jeff Wiegley

Additional Faculty Teaching 400 Level Electives Supporting the Graduate Programs

George Lasik (part-time)

Report Preparation Process

The self-study report was prepared with the significant participation of the Computer Science Department's full-time faculty. Topics, such as the strengths and weakness of the programs, were included on the department meeting agenda and discussed with the full faculty. Then a draft of the text was prepared from the points raised in the discussion and circulated to the faculty. Comments, edits, and additional points were then made by faculty by email and incorporated in the final report. The minutes from the department meetings where these topics were discussed are included in the appendix of the report.

Past and current graduate coordinators contributed to the report's preparation. Ani Nahapetian led the report preparation during the Fall 2012 semester, and then assumed the position of graduate coordinator in the following semester. Robert Lingard, the previous graduate coordinator contributed to the report's preparation. Contributions from the past self-study report, prepared by Richard Lorentz and Steven Stepanek, were used as an important reference in the preparation of this report.

The Department's Industrial Liaison Committee reviewed the document during a May 9, 2013 meeting. They supported the documents analysis of the programs strengths and weaknesses; and suggested some minor changes which were incorporated into the document.

During the report preparation the following strengths were identified: the successful job placement of our graduates, the thesis preparation requirement, and the incorporation of cutting-edge topics in the degree programs. Additionally, we have highlighted several challenges: student technical writing skills, fulfilling desire to grow enrollment, and need for greater resources.

Advisement

All advisement is done through the Department's graduate coordinator. Many students admitted to the program are admitted conditionally classified. These students receive a letter that both encourages them to contact the graduate coordinator for advisement and also lists the undergraduate prerequisite classes they need to take in order to become fully classified. The letter also includes information concerning how long they may expect to spend taking prerequisite classes and other administrative details. Finally, there is reference to a website that deals exclusively with Computer Science Department graduate student matters.

Students admitted fully classified are given a short letter that also encourages them to contact the graduate coordinator and visit the Department's graduate programs website.

The website serves as a major advisement tool. Located at www.ecs.csun.edu/csgrad it includes information about course requirements, undergraduate prerequisite classes, suggestions on how to find a project or thesis topic and how to form a committee, answers to frequently asked questions such as what forms to use when, deadlines, etc., and up to date announcements on matters of significance.

Advisement is currently done in an ad hoc manner, based on feedback from graduate students and discussions with the full faculty. Prior to the last program review, the Department offered a class dedicated to helping students find a thesis topic, prepare a proposal, and deal with all of the necessary paperwork. The faculty decided that advisement through the graduate coordinator and using the website was sufficient, and so the class was discontinued.

Assessment and Strategic Planning

Assessment

A strategic goal of our graduate programs is to offer a curriculum that meets the needs of practicing Computer Science professionals, as well as those who plan to pursue further graduate studies. Our student learning outcomes were developed with this is in mind, and our assessment activities are focused on addressing both the outcomes of greatest concern among the members of our Industrial Advisory Board, as well as satisfying the greatest needs of students wishing to enter Ph.D. programs after graduation.

Data from the U.S. Bureau of Labor Statistics (BLS) supports the need for academic programs in Software Engineering. The U.S. BLS identifies "computer software engineer" as one of the fastest growing job categories in the United States as well as one of the largest growing areas in terms of numbers of people employed. According to BLS data there are currently more software engineers in the U.S. than any other engineering discipline.

Motivated by this information and also advice from our Industrial Advisory Board, the Software Engineering graduate program was added to give students an option for a more focused study of the Software Engineering subfield of Computer Science. While the Computer Science program serves the needs of many students wanting a general education in this academic area and will continue to do so, a growing number of students seek a more comprehensive education in Software Engineering.

Furthermore, there has been a strong demand for more elective courses, especially those in cutting-edge areas of the field. When the new Master's program in Software Engineering was added in 2007, numerous new courses were proposed and added to the Department's graduate course offerings, including Software Requirements, Software Design, Software Project Management, Software Metrics, and Validation and Verification. Also, since the last review period several new 500-level elective courses have been instituted, including Data Mining and Web Engineering. Most recently, a 500-level Mobile Computing experimental course was approved and is planned for a Fall 2013 offering.

Although no specific plan for assessment of the student learning outcomes for our two graduate programs was in place, some assessment activity has been conducted. The extensive assessment program in place for the undergraduate program in Computer Science has yielded assessment data for the graduate programs, as well.

Both undergraduate programs have an outcome that deals with teamwork, and assessment of that outcome was done in classes attended by both undergraduate and graduate students. The undergraduate outcome is, "Function effectively on teams to accomplish a common goal", and the graduate outcome for both programs is, "Work productively in team or collaborative settings to achieve common goals or purposes, including the ability to lead a team."

During both the 2008/2009 and 2009/2010 academic years, teamwork was assessed in 400 and 500 level courses attended by both graduate and undergraduate students using a peer evaluation

questionnaire. The results showed that our graduate students worked productively on teams although no specific evaluation of the ability to lead teams was included in the assessment. The analysis did, however, identify some areas of relative weakness. Specifically, clear communication with other team members and both the "asking for help" and "giving of help when needed" were the areas of relative weakness.

The details of this assessment are described in a paper ("Improving the Teaching of Teamwork in Engineering and Computer Science") by Robert Lingard presented at the International Symposium on Engineering Education and Educational Technologies: EEET 2010 in the context of The 3rd International Multi-Conference on Engineering and Technological Innovation: IMETI in Orlando, FL, June 2010. Two papers by Shan Barkataki and Robert Lingard discuss efforts to improve the teaching of teamwork in project related courses ("Web Based Collaboration in Teaching Teamwork" - EEET 2011 and "Teaching Teamwork in Engineering and Computer Science," - 41st ASEE/IEEE Frontiers in Education Conference).

The challenge is to figure out how the modify the curriculum to provide students with appropriate instruction and opportunity to learn and practice those teamwork skills that were found to be lacking. A recommendation was made that instructors utilizing student teams devise ways to promote communication among team members and reward both the seeking of and the giving of help.

A second assessment activity was undertaken that addressed students' understanding of software engineering principles and practices by administering a test consisting of sample questions from the IEEE Certified Software Development Associate (CSDA) Exam. One of the stated benefits of this exam is for Computer Science or Software Engineering graduates to verify their understanding of Software Engineering principles. This test was administered to graduate students in both Computer Science and Software Engineering in Fall 2010.

For Software Engineering majors this test addressed aspects of many of the learning outcomes but especially the outcome, "Understand software engineering concepts, techniques, practices and tools, and apply them to real problems in a variety of contexts", and for Computer Science majors it addressed many aspects of the outcome, "Demonstrate a knowledge and competence in such fundamental areas of computer science as algorithms, design and analysis, computational theory, computer architecture and software-based systems."

Although preparation opportunity was not provided, most graduate students performed at a level indicating that they would likely pass the real CSDA exam if they took it. This assessment provides, at least, some indication that our graduate students are achieving learning outcomes related to software engineering concepts and practices. A comparison of results between graduate students and senior level undergraduate students who were give the same test show significantly higher scores among graduates as one would hope.

Data regarding the distribution of applicants applying to the Computer Science and Software Engineering degree programs has been maintained at the Department since the last program review. It has shown a large number of international applicants, specifically applying to the Computer Science programs. As a result, the graduate programs website was updated to include

more detailed information for prospective students, and address some of the advisement and recruitment needs of the programs. Information about scholarship and fellowship information was also added to the website, since many international students inquire about funding options. Information regarding the new Software Engineering program was added and enhanced, as well.

The Program's Strategic Plan and its Implementation

In Fall 2012, the assessment of the graduate programs was considered in a meeting with the assistant graduate coordinator and the graduate coordinator, along with interested faculty members. Several assessment plans were developed. Furthermore, the assessment plans were presented to the faculty in a department meeting, where additional suggestions were made.

The following specific assessment tasks have been proposed, to better determine the strengths and challenges of the two Master's programs.

- Classify the completed thesis topics into about 5-10 categories, such as theory, networking, security, graphics, artificial intelligence, etc, to determine the interest-level of students in the various sub-fields of Computer Science. The classification will be carried out by faculty reviewing the thesis titles listed on the Graduate Programs' defense schedule webpage. Information from this assessment task will be used to better inform elective offerings in the Department.

- Alumni information will be maintained using a combination of methods. First, faculty will be asked to report any information they have about their advisee's job or academic placement. Second, data regarding graduate students from the employer surveys, used primarily for undergraduate assessment, will extracted and interpreted. Third, a self-reporting option will be provided for student to inform the department of their current positions.

- To quantitatively assess the quality of writing in the theses produced by students, we plan to have three faculty members read randomly selected theses and score them according to several factors, including survey of literature, quality of writing, and presentation of results. The scores will be used to better inform the faculty about where students need guidance in their thesis preparation.

- An important area of interest for assessment is the question of what parameters signal success in the Master's programs at the time of application. We have maintained high standards for admission into our graduate programs. We plan to analyze the correlation between GRE scores and GPA in the graduate programs, to better determine which parts, if any, should be weighted more heavily in the graduate admissions selection process. For example, we would look for a correlation between GRE scores and graduating GPAs or between GRE scores and program drop-out rates. Also, any differences in graduation rate of students admitted conditionally classified would be relevant.

- We plan to monitor trends regarding enrollment and graduation rates of students from traditionally underrepresented groups. Specifically of interest is the low number of

Hispanic students in the graduate programs, despite their larger presence in the undergraduate programs and their significant numbers in the regions neighboring to the campus. Options for improving the enrollment of local Hispanic students in our Master's program is being considered, including scholarship and federally funded grant opportunities. As part of the planned assessment, we will review the student ethnicity data to see if there is any growth in the enrollment of Hispanic and other traditionally underrepresented student groups in the Computer Science Department.

Discussion of Key Strengths and Challenges

We have highlighted three key strengths of our Master's Programs: the successful job placement of our graduates, the thesis preparation requirement, and the incorporation of cutting-edge topics in the degree programs. Additionally, we have highlighted several challenges: student technical writing skills, fulfilling desire to grow enrollment, and need for greater resources.

KEY STRENGTH –Successful Job Placement of Our Graduates

Both the Computer Science and the Software Engineering Master's Degrees are highly sought after in industry, both locally and nationally. CSUN is located in Los Angeles, which is either near or home to many large and successful companies in the aerospace, entertainment and healthcare industries. Our students receive employment in their field of study, in large numbers, with many students earning starting salaries on the order of $70,000 a year.

The programs' high admission standards coupled with the prerequisite requirements lead to better prepared students. The Master's programs have standards higher than the University requirements for acceptance. We require a GPA of 3.0 or better (as compared with the University minimum of 2.5). Also, we require either GRE scores at or above the 50^{th} percentile on each of the three parts for admission or a Bachelor's degree in Computer Science from an ABET accredited U.S. university. These high standards for admission have led to motivated and well-prepared students.

Due to strong demand for students with skills in these areas, students with degrees outside of Computer Science or Software Engineering apply and gain admission to our programs. We are both willing and have successfully worked with students with academic preparations in different fields. We have a set of undergraduate prerequisite courses that students can bypass if they have completed the courses elsewhere, or they can take the requisite courses at CSUN to become better prepared to succeed in the program.

Faculty relationships with industry partners have also aided students obtain highly-sought after tech jobs. The Department has cultivated a relationship with WellPoint, where selected students obtain training in mainframe systems from IBM and work with WellPoint's systems during their study at CSUN. WellPoint plans to hire the successful students from this program after graduation. Faculty grants from Medtronic have led to projects applying machine learning techniques to diabetic patient data, which has inspired Master's theses in the area. The CECS Honors Coop program, also, provides graduate students with excellent work experience with local companies.

Local companies often contact the Department faculty in an effort to recruit students for competitive job openings. Currently, these openings are advertised informally to students or through posting of fliers. We are exploring creating an email listserve, which students can sign-up for, to distribute information about potential job openings or research opportunities that are targeted to our Department's graduate students.

KEY STRENGTH - Thesis Requirement for Degree

A significant requirement for both Master's degrees is the preparation and completion of a Master's thesis. The work is carried out as part of an independent-study two course series, COMP 696C and COMP 698C, under the guidance of an advisor/committee chair and two other committee members. In the first semester, students work on their original research project, possibly including the development of a software system, carrying out an algorithm design, or performing some data analysis. In the second semester, students prepare and submit the thesis. At the completion of the thesis, students give an oral defense presentation, which is open to all students and faculty to attend.

The thesis preparation process involves the completion of an individual project from start to finish. This includes the analysis of a problem domain; the conception of an idea, usually with significant input from the graduate advisor; the design and implementation of a software system or significant algorithm or the collection and/or development of data; and the completion of some form of analysis of their own work, including but not limited to experimentation, formalization of proofs, preparation of simulation results, and/or completion of user studies.

The thesis work involves a great deal of interaction with several faculty members in the Department and extensive interaction with the student's committee chair. It is not uncommon for the student to interface with the committee chair on a weekly basis. Together they progress through the project work, refining the project requirements, dealing with software development bugs, setting up the necessary experiment plans, and reviewing the written thesis. A significant amount of technical mentoring occurs through these interactions while dealing with the revision of technical text, the development of software, and the aggregation of results. Important soft skills (such as time management, interpersonal interaction with an advisor, status updating) and software engineering skills (such as documentation, version control) are also polished during this process.

The thesis requirement is well aligned with the programs outcomes, as it gives students opportunities to apply the concepts, principles, and practices they have learned and to improve their verbal and written skills. The thesis is usually the largest project and the largest written document that the students have engaged in until that point. Throughout the process, they receive guidance from committee members about their written document and they often see improvements in their writing skills, throughout the process.

The thesis project provides students with the possibility of conducting project and/or research work in an area closely related to the students' current employment. Also, there is a provision for having the industry experts as external members of the student's graduate committee.

Anecdotal evidence has shown that many students keep in touch with their graduate advisor. Some Department faculty keep track of their former advisee by email and by social media (such as Facebook and LinkedIn), as well. This opportunity allows students to grow their professional network, with faculty members and fellow Master's students.

Theses are published by the University and made available through the University library, with the more recent theses available electronically, thanks to the ETD (Electronic Thesis Documentation) system. The availability of a published work is a strong advantage for students seeking competitive industry or research lab job offers. It is also attractive for students applying for Ph.D programs, for inclusion in their admissions applications.

The thesis requirement results in our graduates obtaining expertise in at least one area of study. Therefore in addition to showing potential employers a breadth of knowledge with the variety of coursework taken, students can demonstrate that they are capable of contributing to a specific part of the field and achieving depth in their expertise.

KEY STRENGTH – Incorporation of Cutting-Edge Topics into Program

The nature of the Computer Science and Software Engineering fields is characterized by an ever changing set of principles, tools, and frameworks. Department faculty are not only engaged in reviewing the materials for coursework, but they are also contributing to the field with cutting-edge research.

As an outcome of the growing demand for graduates with expertise in Software Engineering, the Computer Science Department added the Software Engineering Master's program in 2007. As part of the program creation, 7 new graduate courses on Software Engineering were designed and offered.

Students in both programs have the ability to customize graduate programs from a wide selection of, 400, 500 and 600 level classes, to meet specific career goals and learning objectives. Several 600 level core classes are conducted in seminar form, teaching students research and presentation skills, and providing opportunities for learning new and emerging topics in Software Engineering and Computer Science. Additionally, small class sizes also lead to greater access to instructors, with students interacting with the professors directly, not through teaching assistants. The thesis requirement gives faculty and students additional flexibility in working on cutting-edge project areas and subfields, with many innovative topics among the completed theses.

Key Challenge – Student Technical Writing Skills

Limited technical writing experience among our students has been observed, sometimes resulting in delays with the completion of thesis requirement and graduation from the Master's programs. During the preparation of thesis chapters and other written reports, difficulty with written communication has been most pronounced hurdle for our students.

In an effort to deal with this issue, we have added a prerequisite to both programs, English 306: Report Writing. Students with GRE Verbal and/or Analytical Writing scores at or above the 50th percentile are exempted from this requirement.

More recently, a stronger push has been put forward to encourage students to take advantage of the CSUN Writing Center to improve writing, diction, and language use when preparing thesis

chapters. Faculty have been informed of this resource, and its availability is being advertised to students through the graduate programs website and through informal advising.

In terms of admissions standards, we have discussed the utility of using GRE verbal and analytical scores as part of the admissions progress. Currently, students must achieve a score in the 50th percentile or above in each of the three section of the GRE exam. We have discussed using a combination of the verbal and analytical score in our evaluation.

KEY CHALLENGE – Fulfilling Desire to Grow Enrollment

We are eager to grow our enrollment in both Master's programs. The M.S. in Computer Science has had larger enrollments than those seen in recent years. The M.S. in Software Engineering is fairly new, and we believe many professional working in local industry would be eager to join the program.

We are exploring way to improve advertising the programs, e.g. preparing brochures and other marketing materials. We have recently revamped our graduate programs website, as part of this effort.

A significant challenge of growing enrollment is funding for students in the form or research and/or teaching assistantships. Many of our students work in some form or another to support their University costs. If students were provided funding opportunities in their fields of study at CSUN during their education, then we would improve our retention rates, as well as increase graduation and time-to-graduation rates. Also, we could attract more top candidates who view CSUN as a stepping stone to Computer Science Ph.D. programs.

Improved enrollment in our Master's programs would improve the number of graduate courses offered each semester. The lower the number of graduate students enrolled in classes, the less graduate classes offered. Limited course offerings lead to delays in graduation, and in turn discourage new students from joining the program. Additionally, the limited offering of certain graduate electives decreases the attractiveness of our programs, as compared with the diverse course offerings of certain local universities, such as CSU Fullerton.

KEY CHALLENGE – Resources

We have found that resource limitations have detracted from the quality of both graduate programs. Notably, the decreasing numbers of full-time faculty, the growing class sizes, issues with course offerings, hardware and software needs, and the need for a graduate work space have all contributed to serious challenges with our graduate programs.

Faculty staffing levels have been dropping despite growth in student enrollment. Faculty are spread very thin with two graduate degree programs with less resources available than before. We have had several retirements in the last 5 years, with only 2 new faculty hires in that same time frame. In terms of class offerings, FTES and budget restrictions have limited course offerings, while we have seen class sizes grow.

There are several hardware requirements that need to be met to support our graduate programs, both in terms of instruction as well as research. There are limitations on server and software access, including the lack of Perl or Python programming support for web programming thesis projects. Additionally, there is no access to parallel machines for advanced computer architecture courses and research.

Graduate students need work/lab spaces to complete their thesis projects, which is unavailable at the moment. Such space can foster a sense of community and encourage collaboration between students. In the past, we have had dedicated lab space for research, however, we are seeing more instructional use of those spaces. The availability of research labs for faculty with on-going research projects that students can engage in would improve our graduate student experience, as well as encourage faculty research projects and opportunities for external funding.

Discussion of the Department's MOU

An MOU for the Computer Science Master's Program was not generated during the last program review.

There is no MOU for the Software Engineering Master's Program, as the program was instated in 2007 and hence this is the program's first review.

APPENDICES

Program SLOs, Assessment Plan, and Matrices

Student Learning Outcomes of the Graduate Program in Computer Science

Graduates of the Master of Science in Computer Science at California State University, Northridge will be able to:

a. Demonstrate a knowledge and competence in fundamental areas of computer science such as: algorithms, design and analysis, computational theory, computer architecture and software based systems.

b. Demonstrate the analytic skills necessary to effectively evaluate the relative merits of software and computer systems, and algorithmic approaches.

c. Demonstrate a breadth of knowledge in a choice of application areas in computer science, including: networks, artificial intelligence, graphics, human computer interfaces, databases, embedded applications and information security.

d. Understand computer science topics (such as database management, data security, program efficiency, etc.) in a global context (ethics, privacy, human expectations, etc.)

e. Effectively communicate in both written and oral form, especially in areas related to computer science.

f. Work productively in team or collaborative settings to achieve common goals or purposes including the ability to lead a team.

g. Analyze, evaluate and synthesize research and apply theoretical ideas to practical settings.

h. Independently continue their studies in computer science throughout their life.

Student Learning Outcomes of the Graduate Program in Software Engineering

Graduates of the Master of Science in Software Engineering at California State University, Northridge will be able to:

a. Understand software engineering concepts, techniques, practices and tools, and apply them to real problems in a variety of contexts.

b. Define and apply a software process to large-scale real-world problems including requirements analysis and specification, software design and implementation, verification, validation and quality assurance, and the maintenance of software.

c. Analyze and estimate software process costs and manage software development from concept to delivery.

d. Identify, analyze and apply software standards in software engineering practice.

e. Analyze, assess and interpret professional codes of ethics and regulatory documents pertaining to software engineering and understand societal issues.

f. Generate and apply appropriate solutions to solve problems based on reasoned rationale.

g. Work productively in team or collaborative settings to achieve common goals or purposes including the ability to lead a team.

h. Analyze, evaluate and synthesize research and apply theoretical ideas to practical settings.

i. Effectively present ideas, designs and solutions in a logical framework in a variety of forms with proper language structure and mechanics, and to produce appropriate written documentation.

j. Recognize the need for, and show an ability for, dealing with constantly changing technology and continuing professional development.

PROGRAM STUDENT LEARNING OUTCOMES – COMPUTER SCIENCE

PROGRAM REQUIRED COURSES	SLO a	SLO b	SLO c	SLO d	SLO e	SLO f	SLO g	SLO h
610	M	M						
615	M	M						
620	M	D						
630		M						
680				M	D	M		
696C			D		D		D	I
698C			M		M		M	D

PROGRAM STUDENT LEARNING OUTCOMES – SOFTWARE ENGINEERING

PROGRAM REQUIRED COURSES	SLO a	SLO b	SLO c	SLO d	SLO e	SLO f	SLO g	SLO h	SLO i	SLO j
680	D						D	D	D	
682	D	I			I	I	D		D	
684	D	I				I	D		D	
686	D	I	M	D	D	I			D	
696C	D	D				D		D	D	D
698C	M	M				M		M	M	M

I=introduced; D=developed; M=mastered

Course Number: COMP 610

Course Title: Advanced Data Structures and Algorithms

Course Objectives:

A successful student will be able to:

1. Design, optimize, and implement algorithms, with consideration of their computational complexity.
2. Determine and prove the complexity of important classes of problems.
3. Differentiate P, NP-Hard, and NP-complete problems.
4. Develop and prove theoretical bounds for approximation algorithms.
5. Design and analyze online algorithms.
6. Identify several open problems in the field.

Course Number: COMP 615

Course Title: Advanced Topics in Computation Theory

Course Objectives:

A successful student will be able to:

1. Prove closure and non-closure properties for various functions on regular and context-free languages.
2. Prove properties of regular languages using techniques based on deterministic finite automata, nondeterministic finite automata, and regular expressions and understand when which is most appropriate.
3. Prove properties of (deterministic) context-free languages using techniques based on grammars and (deterministic) pushdown automata and understand when which is most appropriate.
4. Show a formal and intuitive understanding of the nature of regular/context-free languages, including being able to differentiate regular/context-free and nonregular/noncontext-free languages, being able to choose the appropriate formalism to describe a language as regular/contest-free or prove it is not regular/context-free, and being able to articulate algorithms for solving problems about regular and context-free languages.
5. Understand the basic terminology concerning Turing Machines including recursive, recursively enumerable, decidable, undecidable, universal Turing Machine, diagonalization, type 0 grammar, generators, enumerators, halting, and non-halting.
6. Be able to prove languages recursive, not recursive, and not recursively enumerable and similarly prove problems decidable and undecidable.
7. Have an intuitive understanding of the differences between recursive and recursively enumerable languages and similarly the differences between decidable and undecidable problems.

Course Number: COMP 620

Course Title: Advanced Computer Architecture

Course Objectives:

A successful student will be able to:

1. Explain innovations in computer hardware design over the last several decades that have resulted in modern single processor commercial platforms: DRAM, chipsets, instruction pipelining, cache memory, and more recently, multicore CPUs.
2. Explain the concept of cache memory in hardware and locality of reference in software, and how computer architectures use these concepts to provide high performance at a low cost.
3. Explain the different types of cache associativity and how it is implemented in hardware.
4. Describe the design spectrum available for computer hardware that supports parallel processing, from shared memory multiprocessors on one end, to local memory multicomputers on the other end.
5. Understand how shared memory multiprocessors support critical section access with mechanisms such as spinlocks.
6. Explain the use of multiple caches and cache coherency policies in a shared memory multiprocessor, and how this use can improve performance by reducing the demand for bus bandwidth.
7. Explain how local memory multicomputers can use message passing to implement communication between processors in lieu of shared memory.
8. Describe common fixed computer network topologies for processor communication: fully connected, line, ring, mesh, hypercube.
9. Describe several examples of multistage interconnection networks (MINs) as a way for processing elements (PEs) to access memory elements (MEs): Omega, Butterfly.
10. Show how existing algorithms designed for single processors can be decomposed to take advantage of parallel processing hardware.
11. Be familiar with common algorithms such as Matrix-Matrix Multiply and Gaussian Elimination are commonly parallelized in both shared memory and local memory environments.

Course Number: COMP 630

Course Title: Formal Semantics of Programming Languages

Course Objectives:

A successful student will be able to:

1. Understand the semantics of abstract programming languages.
2. Identify formal treatments of programming language features, including abstract data types, functions, exceptions, modules, type polymorphism, and objects and classes.
3. Determine if source code from one language can be translated into another language, by analyzing language features.
4. Carryout verification of a program, by providing a formal proof of correctness.
5. Prove simple equivalences between programs using formal semantics, and determine if optimization of code results in a safe transformation of the code.

Course Number: COMP 680

Course Title: Advanced Topics in Software Engineering

Course Objectives:

A successful student will be able to:

1. Understand new and emerging technologies and practices in software engineering.
2. Discuss the relative advantages and disadvantages of new technologies and practices compared with those currently in use.
3. Evaluate and select appropriate technologies and practices with respect to a specific software engineering task.
4. Evaluate how new and emerging technologies and practices impact the field of software engineering.
5. Describe current problems related to the development of large software systems and discuss approaches toward solving them.
6. Demonstrate good communication skills (orally and in writing).

Course Number: COMP 682

Course Title: Software Requirements Analysis and Specification

Course Objectives:

A successful student will be able to:

1. Understand the terminology commonly used in the area of software requirements analysis and specification.
2. Understand the purpose of requirements engineering within the software development lifecycle.
3. Understand the role of requirements engineering within system engineering.
4. Analyze the problems software systems are intended to solve.
5. Communicate with stakeholders to elicit the functional and non-functional requirements for a proposed software system.
6. Define, organize and prioritize requirements for a proposed software system.
7. Develop appropriate use cases and storyboards/prototypes to clarify software requirements.
8. Understand the relationship between software requirements and other development phases (design, implementation and testing), as well as approaches for tracking requirements though these other activities.
9. Produce as part of a team effort a formal software requirements specification.
10. Understand the issues related to the ongoing management of established requirements and changes to them during the course of a software development project.

Course Number: COMP 684

Course Title: Software Architecture and Design

Course Objectives:

A successful student will be able to:

1. Discuss the current challenges in software design.
2. Understand the fundamentals of software design.
3. Construct software designs using appropriate UML diagrams.
4. Understand and discuss the principles of good architectural design
5. Select and apply appropriate architectural patterns and design concepts for the development of a software system.
6. Identify and compare alternate architectural designs for a proposed software system, evaluating the quality of each, and choosing the best one for the problem at hand.
7. As part of a team effort, create and specify a software architectural design for a medium-size software product based on an existing software requirement specification using an accepted program design methodology and appropriate design notation.
8. Understand the principles of detailed design.
9. Apply appropriate design patterns in the detailed design of software.
10. Apply the principles of human-computer interface design appropriate for a given software problem.

Course Number: COMP 686
Course Title: Software Engineering Management
Course Objectives:

A successful student will be able to:

1. Utilize the guiding principles and practices of project management for managing software engineering teams and software projects.
2. Apply and tailor software engineering management models and processes to specific software projects and in specific engineering environments.
3. Explain the importance of planning, documenting, and implementing management plans for software projects.
4. Choose and use resource estimation models (e.g., for schedules, labor hours, equipment, and personnel loading) to plan, budget, control, or bid on software projects.
5. Anticipate the common problems encountered in project management and manage change on software engineering projects.
6. Analyze tradeoffs when selecting software engineering teams, managers, management practices, or project organizations, and understand how these choices may affect the quality, cost and customer acceptance of software projects/products.
7. Discuss current case studies and research trends in software engineering management.

Course Number: COMP 541

Course Title: Data Mining

Course Objectives:

1. A successful student will be able to:
2. Explain the concepts and principles of data mining.
3. Describe the concepts and principles of data warehouse.
4. Demonstrate On-Line Analytical Processing (OLAP) and use a set of OLAP operations.
5. Apply data mining techniques and methods such as data preprocessing, association rules, classification and prediction, and clustering.
6. Be familiar with data mining development software tools and environments currently available on the market.
7. Illustrate data mining applications and trends.

Course Number: COMP 560

Course Title: Expert Systems

Course Objectives:

A successful student will be able to:

1. Understand the nature of knowledge and be able to represent it using the following mechanisms:
   - semantic networks
   - frames
   - first/higher-order predicate calculus
   - lists
   - sets

2. Recognize that PROLOG programs can be viewed as Expert Systems based on the first-order logic.

3. Understand the notion of an Expert System Shell, and be familiar with at least one specific example.

4. Understand the notion of an inference engine, and distinguish between forward and backward chained modes of operation.

5. Understand the notion of a meta-level interpreter, and be able write one in PROLOG for PROLOG.

6. Be able to outline non-standard logics such as many-valued, temporal, modal, order omega, and combinatory.

7. Be able to discuss the issues involved in building an inference engine in a non-standard logic.

8. Understand completely at least one specific Expert System in PROLOG.

9. Know the mechanism of explanation facilities.

10. Be able to outline in PROLOG a simple natural language based interface.

11. Understand the mechanism of self-modifying Expert System.

Course Number: COMP 581

Course Title: Open Source Software Engineering

Course Objectives:

A successful student will be able to:

1. Explain the concepts and principles of Web applications.
2. Apply open source software for developing of a variety of software applications
3. Understand business models using open source software.
4. Understand software license issues and intellectual property rights.
5. Be familiar with open source software products currently available on the market.

Course Number: COMP 584

Course Title: Advanced Web Engineering

Course Objectives:

A successful student will be able to:

1. Explain the concepts and principles of Web applications.
2. Apply current Web technologies for developing Web applications.
3. Be familiar with web application development software tools and environments currently available on the market.
6. Apply Web development processes for developing Web applications.

Course Number: COMP 585

Course Title: Graphical User Interfaces

Course Objectives:

A successful student will:

1. Know and be able to design, develop, and modify software using a current GUI class library (e.g., Java Swing, C# Windows/Web Forms).

2. Know, and be able to describe and contrast the architecture of modern Model / View, event-driven, Object-Oriented GUI class libraries.

3. Know accepted GUI design principles and be able to assess the functional and aesthetic properties of GUI software applications.

4. Know and be able to use models of user performance in GUI tasks that describe, but are not limited to, human error, memory, problem solving, and sensory information processing.

5. Know and be able to use current professional Object-Oriented software development environments.

6. Be able to design, develop, and analyze medium sized software projects involving the design and development of 10 – 20 classes.

7. Know and be able to compare the appropriate use of GUI scripting languages (e.g., Tcl/Tk, Python/TkInter) versus compiled GUI languages.

8. Know and be able to discuss the history, evolution, and contributions of earlier GUI APIs: Smalltalk MVC, Motif / X Windows, and NextStep with Interface Builder.

9. Be able to discuss future trends in GUIs and quickly learn future GUI concepts and APIs.

Course Number: COMP 586

Course Title: Object-Oriented Software Development

Course Objectives:

A successful student will:

1. Have an understanding of the fundamental principles in modeling systems using object-oriented methods.

2. Be able to create system models in Unified Modeling Language (UML) making appropriate use of UML elements for modeling static and dynamic behavior of the system being modeled. (UML).

3. Be able to create system domain models.

4. Be able to create analysis models for software systems.

5. Be able to create design models for software systems at various levels of abstraction to express both static properties and dynamic behavior.

6. Be able to refine and organize the models with packages, components, subsystems, interfaces and ports.

7. Be able to understand and apply common design patterns.

8. Be able to successfully complete a team project that includes: Creation of object-oriented analysis and design models of a moderately complex system in UML and a current CASE tool.

9. Be able to demonstrate the ability to read and analyze computer science literature and synthesize the findings in a term paper.

Course Number: COMP 587

Course Title: Software Verification and Validation

Course Objectives:

A successful student will:

1. Be able to define the common terms used in the area of software verification and validation.

2. Be able to discuss the role of verification and validation in the software development lifecycle.

3. Be able to specify an appropriate testing strategy for given software development activity.

4. Be able to apply appropriate testing techniques to verify and validate software requirements, designs, and implementations.

5. Be able to conduct a formal technical review for a design or implementation.

6. Be able to identify the tasks necessary to accomplish system, regression, performance, stress, and acceptance testing of a software system.

7. Be able to use one or more testing tools to aid in the accomplishment of software verification and validation.

8. Be able to develop a test plan for a given software development project.

9. Be able to conduct a cost/benefit analysis for a planned software testing activity.

10. Be able to produce a formal test plan document, and prepare and deliver an oral presentation of testing results.

Course Number: COMP 589
Course Title: Software Metrics
Course Objectives:

A successful student will be able to:

1. Understand the guiding principles and practices for using software metrics to manage software engineering teams and projects.
2. Apply software metrics in specific software engineering environments.
3. Understand the importance of planning, documenting, and implementing a software metrics program.
4. Understand the common problems encountered in software metrics and measurement and how to avoid them in specific projects.
5. Analyze the tradeoffs when selecting quantitative measurement techniques, practices, or tools, and understand how these choices may affect the quality, cost and customer acceptance of software metrics.
6. Discuss current research trends in software metrics.

Course Number: COMP 598EA

Course Title: Embedded Applications

Course Objectives:

A successful student will be able to:

1. Evaluate software engineering principles for application on specialized hardware.
2. Integrate sensor data and actuators to allow software to interact with physical environments.
3. Employ specialized architectures to handle response needs of embedded devices.
4. Discuss the difference between real-time and classical operating systems.
5. Construct software capable of overcoming hardware limitations.
6. Research materials necessary to understand foreign hardware.
7. Interact successfully with other engineers to produce functional physical products.
8. Design fault tolerant software programs.
9. Criticize asynchronous systems in the presence of faults to produce possible causes of error.

Course Number: COMP 598NSP

Course Title: Advanced Network Security Projects

Course Objectives:

A successful student will be able to:

10. Construct a functioning TCP/IP network using ethernet components.

11. Demonstrate an ability to locate needed resources.

12. Operate a variety of network applications to provided e-commerce level service to clients.

13. Investigate the cause of system and network anomalies.

14. Design plans for the protection and repair of network security threats.

15. Defend network decisions within a corporate and economic environment.

16. Debate the merits of particular solutions for effectiveness and security.

Course Number: COMP 410

Course Title: Logic Programming

Course Objectives:

A successful student will:

1. Understand that logic programming in general is based on formal logic, and that computation is in effect an application of a proof procedure.
2. Know the syntax of PROLOG.
3. Understand the declarative and operational semantics of PROLOG.
4. Know the fixed-point theorem.
5. Be able to argue as to why PROLOG is a universal language.
6. Know the notion of positive world assumption.
7. Understand as to why negation is not computable.
8. Know how to implement in PROLOG negation by failure.
9. Know and be able to use the mechanism of the cut in PROLOG.
10. Be able to distinguish the three types of equalities in PROLOG: identity, expression value, and unification.
11. Understand the notion of program interpreter.
12. Be able to outline a Prolog interpreter.
13. Know the implementation of lists in PROLOG.
14. Be able to define in PROLOG basic list functions.
15. Understand the meta predicates assert and retract.

Course Number: COMP 421

Course Title: The Unix Environment for Programmers

Course Objectives:

A successful student will be able to:

1. Understand and utilize various shells in the software development process.
2. Understand and utilize common UNIX tools, e.g., sed, awk, grep, and demonstrate the ability to develop composite tools using shell-programming technics.
3. Demonstrate the ability to use current Source Control Systems used in program development.
4. Demonstrate the ability to use Software Configuration Management tools, such as autoconf, makefiles, etc.
5. Demonstrate knowledge of concurrence, scheduling, task coordination, and deadlock prevent using POSIX pthreads.
6. Demonstrate the ability to utilize Interprocess Communication, Network Sockets, and Distributed Computing concepts via the construction of an application using the client-server model.

Course Number: COMP 424

Course Title: Computer System Security

Course Objectives:

A successful student will be able to:

1. Communicate about computer security through oral and written reports and group discussions.
2. Understand the functions and relationships of computer system security systems.
3. Understand and analyze security encryption mechanisms and be able to cite the strengths and weaknesses of the various encryption algorithms, and other security issues surrounding them.
4. Understand the function of security controls such as firewalls, authentication, and intrusion detection devices.
5. Identify the vulnerabilities of computer systems, and how they can be protected.
6. Understand the ethical and legal issues that arise from new technology related to computer security.

Course Number: COMP 426

Course Title: Fault-Tolerant Software and Computing

Course Objectives:

A successful student will be able to:

1. Explain the need for fault tolerance in software-intensive systems, especially safety-critical, hard real-time, or networked transaction-based applications.
2. Describe the basic concepts and measures of fault tolerance, such as fault, failure, redundancy, single point of failure, hard/soft/Byzantine failures, MTTF, reliability, availability, and maintainability.
3. Analyze standard hardware fault tolerant computing systems and techniques, such as RAID, voters, network topologies, M-of-N systems, watchdog timers, and parity bits.
4. Use software fault tolerant techniques (e.g., N-version programming, recovery blocks, acceptance tests, checkpointing) to design reliable computing systems.
5. Implement information redundancy in software, including the common error detection/correction codes (e.g., checksums, M-of-N, Berger, cyclic, arithmetic).
6. Apply simulation and experimental tools like CARMS to statistically measure system reliability.
7. Employ software reliability models to estimate software defect rates, predict software reliability, or determine when to ship software.
8. Locate and critically evaluate recent advanced resources (e.g., articles, books, tools, case studies) on software fault tolerance.

Course Number: COMP 429

Course Title: Computer Network Software

Course Objectives:

A successful student will be able to:

1. Know the history and evolution of the TCP/IP Communications Suite.
2. Diagram UDP and TCP communication.
3. Describe the differences between circuit switched and packet switched networks.
4. Discuss the ISO 7-layer and TCP/IP 5-layer models.
5. Produce server/client applications using the Berkeley sockets interface.
6. Understand the major protocols and data structures used for the communication layers of the TCP/IP model.
7. Discuss security issues related to network communication.
8. Explain an application-level protocol in functional detail.

Course Number: COMP 430

Course Title: Language Design and Compilers
Course Objectives:

A successful student will be able to:

1. Describe the core components of a compiler, understand the role of lexical analysis, syntax analysis, intermediate code generation, and code generation. Be familiar with various optimization technics that are programming-language and target-machine independent.

2. Understand the functions and relationships of compilers and compilation systems.

3. Understand and apply the concepts of language theory with an emphasis on Regular and Context Free Languages.

4. Understand the semantics of programming languages via operational semantics, and be able to apply this understanding via the translation of various programming constructs.

5. Understand and apply knowledge of data structures and program design via the construction of a compilation system that includes symbol tables, parser trees, intermediate code representation, etc.

6. Understand the versatility of using and building tools to add in the software development process.

Course Number: COMP 432

Course Title: Object Oriented Programming

Course Objectives:

A successful student will be able to:

1. Compare, understand and use in the design and implementation of software the concept of Abstract Data Types (ADT) as defined in several current Object Oriented Programming Languages. Depending on the language, these ADTs would include: class, interface, generics, delegate, property, and event.

2. Understand and use basic Object Oriented principles: information hiding, encapsulation, instance and class variables, aggregations (composition), inheritance (single, multiple, behavioral), polymorphism, meta-classes, and reflection.

3. Compare, understand and use current Object Oriented Design techniques such as: CRC (Component Responsibility Collaborator), Design Patterns, and UML (Unified Modeling Language).

4. Compare, understand and use class libraries from current Object Oriented Languages for standard development tasks (e.g., collections, graphical user interfaces for desktop and web applications, network database connection and query, security).

5. Be able to implement representative software solutions incorporating 5 – 10 abstract data types, in current object oriented programming languages (e.g., C#, Java, Python, Squeak,).

6. Know and be able to discuss source code translation issues pertaining to Object Oriented Languages in comparison to non-OOP languages.

7. Know and be able to discuss the history, evolution, and contributions of earlier Object Oriented Programming languages: Simula, Smalltalk, and C++.

8. Be able to discuss future trends in OOP languages and quickly learn future OOP concepts and languages.

Course Number: COMP 440

Course Title: Database Design

Course Objectives:

A successful student will be able to:

1. Understand the normal forms.
2. Be able to create a web-database application. Learn how to follow the instruction for implementation.
3. Be able to apply appropriate testing techniques to verify and validate database software requirements, designs, and implementations.
4. Be able to conduct a formal technical (functional dependencies) review for a design or implementation. Understand the role of functional dependencies and the design based dependencies.
5. Be able to identify the tasks necessary to accomplish a 3 tier system.
6. Be able to develop a test plan for a given software development project.
7. Be able to produce a formal document including source code, database script, and implementation, and prepare and deliver an oral and demo presentation of project.
8. Be able to participate in a team project.

Course Number: COMP 465

Course Title: Computer Graphic Systems and Design

Course Objectives:

A successful student will:

1. Know and be able to describe the general software architecture of programs that use 3D computer graphics.
2. Know and be able to discuss hardware system architecture for computer graphics. This includes, but is not limited to: graphics pipeline, frame buffers, and graphic accelerators/co-processors.
3. Know and be able to use a current 3D graphics API (e.g., OpenGL or DirectX).
4. Know and be able to use the underlying algorithms, mathematical concepts, supporting computer graphics. These include but are not limited to:
    - Composite 3D homogeneous matrices for translation, rotation, and scaling transformations.
    - Plane, surface normals, cross and dot products.
    - Hidden surface detection / removal.
    - Scene graphs, display lists.
5. Know and be able to select among models for lighting/shading: Color, ambient light; distant and light with sources; Phong reflection model; and shading (flat, smooth, Gourand, Phong).
6. Know and be able to use and select among current models for surfaces (e.g., geometric; polygonal; hierarchical; mesh; curves, splines, and NURBS; particle.
7. Know and be able to design and implement model and viewing transformations, the graphics pipeline and an interactive render loop with a 3D graphics API.
8. Be able to design and implement models of surfaces, lights, sounds, and textures (with texture transformations) using a 3D graphics API.
9. Be able to discuss the application of computer graphics concepts in the development of computer games, information visualization, and business applications.
10. Be able to discuss future trends in computer graphics and quickly learn future computer graphics concepts and APIs.

Course Number: COMP 467
Course Title: Multimedia System

Course Objectives:

A successful student will be able to:

1. Demonstrate knowledge of the foundations, techniques, limitations, and applications of multimedia computing applications.
2. Describe the mathematical principles of underlying data compression algorithms.
3. Recognize the current technical issues facing multimedia software developers when managing multimedia data.
4. Understand the factors involved in multimedia application's performance, integration and evaluation.
5. Apply theories and research.

Course Number: COMP 469

Course Title: Introduction to Artificial Intelligence

Course Objectives:

A successful student will be able to:

1. Understand the nature of intelligence and its manifestations in activities such as:
   - the use of sound and complete deductive logic
   - creativity
   - intuition
   - learning from experience
   - natural language understanding
   - pattern recognition
2. Understand and use the language of the first-order predicate calculus to represent logical propositions.
3. Understand that the notion of validity represents universal truth.
4. Understand and be able to apply Herbrand's algorithm for determining validity.
5. Recognize PROLOG as a subset of the first-order predicate calculus.
6. Construct in PROLOG programs which demonstrate the use of sound and complete deductive logic.
7. Understand searching techniques, such as depth-first and breath-first, and be able to use them for creating logic programming interpreters.
8. Understand the notion of meta-interpreter.
9. Use PROLOG to design and construct meta-interpreters.
10. Understand the notions of forward and backward chaining interpreters and be able to use them to simulate deductive and inductive reasoning.
11. Understand the relation between inductive reasoning and creativity and intuition; be able to use PROLOG to simulate these activities.

Course Number: COMP 484

Course Title: E-Business Technologies

Course Objectives:

A successful student will:

1. Have an understanding of the Internet architecture- the Internet backbone, domain name translation architecture; computer networking principles and protocols used in the Internet.

2. Understand the principles of creating user friendly, cross browser capable, and ADA compliant websites using the prevailing website creation technologies.

3. Have an understanding of the web hosting technologies including planning, installation, and security issues related to modern web servers.

4. Understand the principles of design, implementation and testing of client side web applications.

5. Understand the principles of server side programming, including the use of database servers and session management.

6. Have an understanding of the use of data exchange languages such as, XML and the related technologies.

7. Understand the internet security risks and the principles aimed at mitigating such risks.

8. Be able to demonstrate the ability to read and analyze computer science literature and synthesize the findings in a term paper.

Course Number: COMP 485
Course Title: Human-Computer Interaction

Course Objectives:

A successful student will be able to:

1. Demonstrate an understanding of guidelines, principles, and theories influencing human computer interaction.

2. Recognize how a computer system may be modified to include human diversity.

3. Select an effective style for a specific application.

4. Design mock ups and carry out user and expert evaluation of interfaces.

5. Carry out the steps of experimental design, usability and experimental testing, and evaluation of human computer interaction systems.

6. Use the information sources available, and be aware of the methodologies and technologies supporting advances in HCI.

Faculty Vitae

1. Name: **Jack D. Alanen**

2. Education

- Ph.D. Statistics, Yale University, 1972
- M.Ph. Statistics, Yale University, 1967
- M.S. Statistics, Yale University, 1966
- M.S. Mathematics, Case Institute of Technology, 1962
- B.S. Mathematics, Case Institute of Technology, 1960

3. Academic experience – institution, rank, title, when, and full- or part-time

- CSUN Computer Science, Full Professor, 2011-2016, FERP
- CSUN Computer Science, Full Professor, 2002-2011, full-time
- CSUN College of Engineering and Computer Science, Associate Dean, 2000-2002, full-time
- CSUN Computer Science, Lecturer , 1990-2000, part-time
- UCLA Extension, Computer Science, Lecturer,1990-2000, part-time
- CSU, Long Beach, Computer Science, Lecturer, 1995-1998, part-time
- CSUN, Computer Science, Full Professor, 1980-1983, full-time
- CSUN, Computer Science, Department Chair, 1980-1982, full-time
- SUNY at Buffalo, Computer Science, Associate Professor, 1975-1976, full-time
- University of Nairobi, Kenya, Computing Center Department, Head, 1974-1975, full-time
- University of Connecticut, Storrs, Statistics, Assistant Professor, 1968-1969, full-time

4. Nonacademic experience

- Litton Systems, Inc., 1990-2000 and 1982-1988; Senior Scientist, Litton Data Systems (1998-2000); Scientist, Litton Data Systems (1993-97); Senior Member Technical Staff, Litton Aero Products (1990-92); Director of Software Engineering, Litton Aero Products (1986-88); Member Technical Staff, Litton Guidance & Control Systems (1983-85); Consultant at Litton Guidance & Control Systems Division (1982-83).
- SoHaR (Software Hardware Reliability) Inc., Senior Research Engineer, Supported the FAA and IBM on the Advanced Automation System (modernization of the air traffic control system), especially in the areas of reliability, maintainability, and availability, 1988-1990, full-time
- Automatic Test Equipment Solutions, Inc., Consultant - Design and teach courses in software engineering to government and industry personnel for various organizations, e.g., U.S. Army, U.S. and Turkish Air Forces, and Allied-Signal Corp., 1983-present, part-time
- Case Western Reserve University, Director of Academic Computing, A. R. Jennings Computer Center, 1978-1979
- University of Utrecht, The Netherlands, Senior Staff Member, Informatics Department, 1976-1978

- Mathematical Center, Amsterdam, The Netherlands, Scientific Collaborator, Computing Department, 1970-1974
- Yale University, Manager of Operations, Yale Computer Center, 1964-1965
- IBM Advanced Systems Division, Associate Programmer, 1963-1964

5. Certifications or professional registrations

- None

6. Current membership in professional organizations

- ACM; IEEE; IEEE Computer Society, American Statistical Association; Society for Sigma XI; American Association of University Professors; California State University Emeritus & Retired Faculty Association

7. Honors and awards

- None

8. Service activities (within and outside of the institution)

- CSUN, Computer Science, Graduate Coordinator & Transfer Advisor, College representative to the University PP&R Committee; Thesis advisor for Radhika Veldurthi, MS thesis: *Real Time and Interactive Analysis of Database Query Reports*.
- United States Power Squadrons (USPS, a nonprofit organization): Special Assistant to the Budget Director, National Office; Webmaster and Newsletter Editor for a local squadron (Alamitos Sail and Power Squadron, Huntington Beach, CA).

9. Recent publications and presentations

- "Comparing Software Design for Testability to Hardware DFT and BIST," by Jack Alanen and Louis Ungar, refereed paper presented at IEEE AUTOTESTCON 2011, Baltimore MD, September 12-15, 2011, and published in IEEE Systems Readiness Technology Conference 2011 Proceedings.

10. Recent professional development activities

- Reviewer for McGraw-Hill for new book on Information Security by Shimeall and Veltsos, 2013
- Reviewer for McGraw-Hill of the book *Programming Languages: Principles and Paradigms*, 3rd edition, by Tucker, 2012
- Invited participant at McGraw-Hill's *Microsoft Office 2012: In Practice Symposium*, Pasadena, CA, April 19-22, 2012
- Attended *Course Technology Forums 2012*, sponsored by Cengage Learning, Newport Beach, CA, March 16, 2012
- Sabbatical at Buraimi College, Oman; assisted the Dean of the College with development of their curriculum and various academic programs, Fall 2007

1. Name: **Prasanta (Shan) Barkataki**

2. Education
   - Ph.D. Computer Science, Univ. of Bradford, England, 1976

3. Academic experience
   - CSUN, Full Professor, 1981-present, full-time
   - Teesside University, England, Lecturer, 1973-1977, full-time

4. Non-academic experience
   - Systems & Programming Manager, Comp Center, Teesside University, 1977-1981, full-time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM-SigAda, SIGITE

7. Honors and awards
   - Engineer's Council Distinguished Engineering Educator award, 2013
   - $143,000 grant from WellPoint Inc. to develop courseware on Enterprise Computing.
   - $68000 in externally funded cash grants and Design Clinics from Litton Data Systems, Northrop Grumman Corp, and Online Administrators.
   - Two time recipient of Julian Beck award.
   - Multi-year grant from Microsoft Corporation providing free use of Microsoft software by students and faculty at home and in college labs.
   - Numerous software license grants from software development tools from companies such as IBM, Rational Corporation, Visual Paradigm, AdaCore technologies, and Altova corp.

8. Service activities (within and outside of the institution)
   - CSUN, Academic Technology Committee
   - CSUN, College Dean Search Committee
   - CSUN, Department personnel committee (Chair)
   - CSUN, Department Faculty Search Committee
   - CSUN, Department PTR Evaluation Committee (Chair)
   - CSUN, University Extended Learning
   - CSUN, College Academic Affairs (Chair)
   - CSUN, College Research & Services
   - CSUN, Academic Senate
   - CSUN, College Resource Committee
   - CSUN, College Scholarship Committee (Chair)

9. Recent publications and presentations
    - "Distributed Container: A Design Pattern for Fault Tolerance and High Speed Data Exchange"; Shan Barkataki et. Al; Proc. ACM's Annual International Conference on Ada and Related Technologies**,** St. Petersburg, FL, 2009
    - "Process Streamlining with Entropy Reduction"; Shan Barkataki et. Al; IEEE/INCOSE/DOD Software and System Technology conf, Salt Lake City, UT, 2009
    - "Software Reengineering with SOA and Web Services"; Shan Barkataki et. Al; IEEE/INCOSE/DOD Software Technology Conference, Salt Lake City, UT, 2009.
    - "Empowering Legacy Reuse with Service Oriented Architecture and Web Services"; Mike Vertuno, Shan Barkataki; NGC SEAG Conf, Redondo Beach, CA, 2008

10. Recent professional development activities
    - 1 week course on Enterprise Computing for Academics, IBM innovation center, Dallas, TX, Aug, 2012
    - Enterprise Computing Conference, Poughkeepsie, NY, June 2012
    - SigAda conference, 2009
    - Software Technology Conference, Salt Lake City, 2009
    - SIGITE conference, Cincinnati, 2008

1. Name: **George Michael Barnes**

2. Education
   - Ph.D. Experimental Psychology, University of Kansas, 1980
   - M.S. Computer Science, Kansas State University, 1980
   - M.A. Psychology, California State University, Long Beach, 1975
   - A.B. Psychology, University of California Berkeley, 1972

3. Academic experience
   - CSUN Computer Science, Full Professor, 1989-present
   - CSUN Computer Science, Associate Professor, 1984-1989
   - CSUN Computer Science, Assistant Professor, 1981-1984
   - Kansas State University, Assistant Professor, 1980-1981

4. Non-academic experience
   - Consulting various companies, software development and instruction, part time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - Association for Computing Machinery
   - Consortium for Computing Sciences in Colleges, Southwestern region

7. Honors and awards
   - None

8. Service activities (within and outside of the institution)
   - CSUN, Department recruitment committee
   - CSUN, Department personnel committee
   - CSUN, Department academic program improvement committee.

9. Recent publications and presentations
   - Using an Instructor Authored Visual Simulation Framework in a CS3 Course, Accepted for presentation April 2013 and subsequent publication, Journal of Computing Sciences in Colleges**.**
   - SceneWorld: A Game-Oriented Graphics Course Starter Kit, *The* Journal of Computing Sciences in Colleges, April 2009, 24, 4, pp 268-274.

10. Recent professional development activities
    - Self-study, software development, and conference attendance

1. Name: **Richard Covington**

2. Education
   - Ph.D. Electrical and Computer Engineering, Rice University, 1989
   - M.S. Electrical Engineering, Rice University, 1985
   - B.A. Physics, Harvard University, 1980

3. Academic experience
   - CSUN Computer Science, Full Professor, -present, full-time
   - CSUN Computer Science, Associate Professor, 2006-, full-time
   - CSUN Computer Science, Assistant Professor, 2000-2006, full-time

1. Name: **Steven M. Fitzgerald**

2. Education
   - D.Sc. Computer Science, University of Massachusetts, Lowell, 1994
   - M.S. Computer Science, University of Massachusetts, Lowell, 1990
   - B.S. Computer Science, University of Massachusetts, Lowell, 1986

3. Academic experience
   - CSUN Computer Science, Associate Professor & Director, Pioneering Technology Group, 1999 – Present, full-time
   - CSUN Information Technology Research, Chief Technology Officer, Jan. 2001 – Jun. 2001, full-time
   - CSUN Information Technology Research, Information Security Officer, Jan. 2001 – Jun. 2001, 50% Faculty release-time
   - CSUN Computer Science, Assistant Professor, 1994 – 1999, full-time
   - USC Information Science Institute, Computer Scientist, 1996 – 2001, full-time

4. Non-academic experience
   - Eucalyptus Systems, Inc, VP, Technical Services, Built and developed a team for Customer Support, IT, and Web Operations, 2010 – 2012, full-time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - None

7. Honors and awards
   - Author of two paper selected for the Top Twenty Papers in the First Twenty Years of HPDC, 2012
   - Previously held the World Record for the "Largest, distributed, interactive simulation", 1998
   - Received "Best of Show" award at the 1998 Super Computing Conference for most innovative wide-area application on a "Computational Grid"
   - Received "Meritorious Service Award" from USC for the demonstrating the largest "Computational Grid" which include over 3,000 data processor spread throughout in the U.S and Europe, 1997

8. Service activities (within and outside of the institution)
   - Academic Affairs Web Steering Committee
   - Class Room Technology Committee
   - Technical Infrastructure and Services Committee

9. Recent publications and presentations
    - Provost Professional Development Series on the Virtual University, Spring 2008
    - "Myths and Facts about Accessibility," at the 2008 Department Chairs and Deans Retreat, Fall 2008

10. Recent professional development activities
    - None

1. Name: **Peter Gabrovsky**

2. Education
   - Ph.D. Computer Science, Syracuse University, 1977
   - M.S. Computer Science and Mathematics, Warsaw University, Poland, 1968
   - Undergraduate degree, Sofia University, Sofia, Bulgaria, 1965

3. Academic experience
   - CSUN Computer Science, Full Professor, 2009-present, full-time
   - CSUN Computer Science, Associate Professor, 1992-2009, full-time
   - CSUN Computer Science, Assistant Professor, 1989-1992, full-time
   - University of Southern Maine, Computer Science, Associate Professor, 1985-1989, full-time

4. Non-academic experience
   - United Missouri Bank, Executive Vice-president, Kansas City, MO, 1979- 1985, full-time
   - IBM, Senior Associate Programmer, 1969-1979, full-time
   - Programmer, PX European Headquarters, Germany, 1968-1969, full-time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - Member of the Consortium for Computing Sciences in Colleges (CCSC)

7. Honors and awards
   - None

8. Service activities
   - CSUN, Personnel, Recruitment, and Chair search committees
   - Author Chair of CCSC/SW (three year term ending 2014)
   - Co-chair of CCSC/SW 2014
   - Reviewer for the professional journals “Studia Logica” and “Modern Logic”, where I also served as an Editor
   - I have served as a text book reviewer for several publishers. One of these textbooks, “Computer Science: A Structured programming Approach Using C”, by Gilberg and Forouzan, has been very successful, and is presently in its third edition
   - I have also made several presentations on the subject of Artificial Intelligence at local high schools, the CSUN chapter of the Association for Computing Machinery, and for prospective freshmen here at CSUN

9. Recent publications and presentations

- “EXLOG: a Non-standard Logic Programming Language for Experiment-based Research” - scheduled for presentation at the 14th CLMPS; “Logic Programming with Generalized Quantifiers” – a paper published in the proceedings of the CCSC/SW 2010
- “Extending Logic Programming Beyond Computability” - a paper published in the proceedings of the CCSC/SW 2009
- "Teaching Methodology of Artificial Intelligence and Related Subjects: Meeting Industry's Needs" – participant in a panel at the CCSC/SW 2010 conference
- “LABLOG: A Non-standard Logic Programming Language for Experiment-base Research” – a scheduled presentation at the International Conference on Computer Science and Information Systems, Athens, 2007

10. Recent professional development activities
    - 6th IASTED Conference on Human Computer Interaction, May 16-18, 2011
    - Conferences of CCSC/SW in 2010, 2011, and 2012

1. Name: **Adam Kaplan**

2. Education
   - Ph.D. Computer Science, UCLA, 2008
   - M.S. Computer Science, UCLA, 2003
   - B.S. Computer Science and Engineering, UCLA, 2000

3. Academic experience
   - CSUN, Computer Science, Assistant Professor, 2012-present, full time
   - CSU, Dominguez Hills, Computer Science, Instructor, 2009-2012, part time
   - UCLA, Computer Science, Instructor, 2010-2012, part time
   - UCLA Extension, Engineering, Instructor, 2009-2009, part time

4. Non-academic experience
   - Perceptive Development, Inc., iOS Application Developer and Legal Consultant, developed iPhone and iPad applications and reviewed third-party source code on behalf of an expert witness in a patent lawsuit, 2009-2012, part/full time
   - IEnteractive Research and Technology, Research Scientist, designed physics benchmark suites and graphics pipelines for high-performance multi-core machines and submitted grant proposals to funding agencies, 2008-2009, full time
   - Telephone Unlimited, PHP Developer and Database Designer, created user registration utility and CRM software, 2003-2003, part time
   - AnswerFinancial Inc., Enterprise Integration Application Developer, developed translation programs to convert raw flat data to XML and relational representations, 2001-2001, full time
   - National Information Consortium, Java Developer, designed small-government software for municipalities, 2001-2001, full time
   - Styleclick, Inc., Java Developer, designed and implemented auction website, 2000-2001, full time
   - drDrew.com, Web Programmer, wrote content delivery, administrative, and graphics-generation web software, 2000-2000, full time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - American Society For Engineering Education (ASEE)

7. Honors and awards
   - Outstanding Engineering Achievement Merit Award, CSUN College of Engineering & Computer Science, 2013
   - Best Paper, IEEE International Symposium on High-Performance Computer Architecture (HPCA), February 2008

8. Service activities (within and outside of the institution)
    - Member of Department Improvement Committee, CSUN Computer Science Department

9. Recent publications and presentations
    - M. Frank Chang, Jason Cong, Adam Kaplan, Mishali Naik, Glenn Reinman, Eran Socher, Sai-Wang Tam, "CMP Network-on-Chip Overlaid With Multi-Band RF-Interconnect," IEEE International Symposium on High-Performance Computer Architecture (HPCA*)*, February 2008
    - Jason Cong, Karthik Gururaj, Guoling Han, Adam Kaplan, Mishali Naik, and Glenn Reinman, "MC-Sim: An Efficient Simulation Tool for MPSoC Designs," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), November 2008
    - M. Frank Chang, Jason Cong, Adam Kaplan, Chunyue Liu, Mishali Naik, Jagannath Premkumar, Glenn Reinman, Eran Socher, and Sai-Wang Tam, "Power Reduction of CMP Communication Networks via RF-Interconnects," 41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), November 2008

10. Recent professional development activities
    - Attended Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), March 2011

1. Name: **Robert W. Lingard**

2. Education
   - Ph.D. Computer Science, USC, 1975
   - M.S. Computer Science, USC, 1970
   - M.A. Mathematics, UCLA, 1965
   - B.A. Mathematics, UCLA, 1963

3. Academic experience
   - CSUN, Computer Science, Full Professor, 2012-present, FERP
   - CSUN, Computer Science, Full Professor, 2007-2012, full-time
   - CSUN, Computer Science, Associate Professor, 2001-2007, full-time
   - CSUN, Computer Science, Assistant Professor, 1996-2001, full-time
   - CSUN, Computer Science, Lecturer, 1994-1996, full-time
   - CSUN, Computer Science, Lecturer, 1979-1984, part-time
   - USC, Lecturer, 1975-1977, part-time

4. Non-academic experience
   - Advanced Computing Systems Company, Director of Development, Managed development and research related to systems software, 1991-1993, full-time.
   - CADAM INC, An IBM Co, Director/Researcher, Managed development and research related to CAD/CAM software, 1981-1991, full-time.
   - USC/Information Sciences Institute, Member of the Research Staff, Conducted artificial intelligence research, 1978-1981, full-time.
   - Lockheed-California Company, Programmer/Supervisor/Researcher, Developed software, managed software development and conducted software applications research, 1963-1978, full-time.

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM; IEEE Computer Society; ASEE

7. Honors and awards
   - None

8. Service activities (within and outside of the institution)
   - CSUN, Member of the Faculty Senate; Assessment Liaison for the department on the University's Assessment Committee (AALC); Chair of the College Assessment Committee; Department Assessment Coordinator; Member of the Department's Program Improvement Committee

9. Recent publications and presentations
    - "Teaching Teamwork in Engineering and Computer Science," with Shan Barkataki, Proc. 41st ASEE/IEEE Frontiers in Education Conference, Rapid City, SD, October 2011.
    - "Web Based Collaboration in Teaching Teamwork," with Shan Barkataki, Proc. International Symposium on Engineering Education and Educational Technologies: EEET 2011 in the context of The 4th International Multi-Conference on Engineering and Technological Innovation: IMETI, Orlando, FL, July 2011.
    - "Improving the Teaching of Teamwork in Engineering and Computer Science," Proc. International Symposium on Engineering Education and Educational Technologies: EEET 2010 in the context of The 3rd International Multi-Conference on Engineering and Technological Innovation: IMETI, Orlando, FL, June 2010.
    - "Analyzing Reliability and Validity in Outcomes Assessment," with Deborah Van Alphen, presented at the ABET Symposium, Las Vegas, NV, April 2010.
    - "Teaching and Assessing Teamwork in Engineering and Computer Science," Proc. International Symposium on Engineering Education and Educational Technologies: EEET 2009 in the context of The 2nd International Multi-Conference on Engineering and Technological Innovation: IMETI, Orlando, FL, July 2009.
    - "Collecting and Interpreting Quantitative Data," with Deborah Van Alphen, presented at the Annual Western Assessment Conference XIII, Fullerton, CA, March 2009.
    - "Assessing Student Learning in Software Engineering," with Taehyung Wang and Diane Schwartz, Journal of Computing Sciences in Colleges, Vol. 23, No. 6, June 2008.
    - "Service Learning in Math and Computer Science: Panel Discussion," with Lori Carter, Erin Trine, Nathan Hirst and Catherine Marcarelli, Journal of Computing Sciences in Colleges, Vol. 23, No. 6, June 2008.

10. Recent professional development activities
    - CSUN: ABET Assessment Workshop, 2012; ASEE/IEEE Frontiers in Education Conference, 2011; International Symposiums on Engineering Education and Educational Technologies (EEET), 2012, 2010, 2009; ABET Symposium, 2010; Annual Western Assessment Conference, 2009; Consortium for Computing Sciences in Colleges, Southwestern Conference, 2008

1. Name: **Richard Lorentz**

3. Education
   - Ph.D. Computer Science, Washington State University, 1980
   - B.A. Mathematics, Claremont McKenna College, 1975

3. Academic experience
   - CSUN, Assistant/Assocate/Full Professor, 1987 – present
   - Harvey Mudd College, Assistant Professor, 1983 – 1987
   - Claremont McKenna College, Assistant Professor, 1982 – 1983
   - New Mexico State University, Assistant Professor, 1980 – 1982

4. Non-academic experience
   - None

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM, International Computer Games Association

7. Honors and awards
   - 2010 – 2011 College of Engineering and Computer Science Research Fellow

8. Service activities (within and outside of the institution)
   - CSUN Computer Science Department Graduate Coordinator, 2007 – 2009
   - Program Committee member for the International Conference on Computers and Games, 2013, 2012, 2011, and 2010
   - Program Committee for the Conference on Technologies and Applications of Artificial Intelligence, 2011 and 2010
   - Program Committee for the Second International Workshop on Computer Game and its Applications, 2012
   - External Referee for doctoral theses (2), Friedrich Schiller University of Jena, Germany, 2009 and 2010

9. Recent publications and presentations
   - *An MCTS Program to Play EinStein Wurfelt Nicht!*, in In H. Jaap van den Herik, A. Platt, editors, Computers and Games, 8th International Conference, CG2011, volume 7168 of Lecture Notes in Computer Science, pages 52-59, Springer-Verlag New York, Inc., 2011

- *Improving Monte-Carlo Tree Search in Havannah.* In H. Jaap van den Herik, H. Iida, A. Platt, editors, Computers and Games, 7th International Conference, CG2010, volume 6515 of Lecture Notes in Computer Science, pages 105-115, Springer-Verlag New York, Inc., 2010
- *Amazons Discover Monte-Carlo.* In H. Jaap van den Herik, X. Xu, Z. Ma, M. Winands, editors, Computers and Games, 6th International Conference, CG2008, volume 5131 of Lecture Notes in Computer Science, pages 13-24, Springer-Verlag New York, Inc., 2008.

10. Recent professional development activities
    - Attended at competed various programs in the last five Computer Olympiads held in Tilburg, The Netherlands; Kanazawa, Japan; Pamplona, Spain; Beijing, China; and Amsterdam, The Netherlands.

1. Name: **Robert McIlhenny**

2. Education
   - Ph.D. Computer Science, UCLA, 2002
   - M.S. Computer Science, UCLA, 1996
   - B.S. Information and Computer Science, UCI, 1993

3. Academic experience
   - CSUN Computer Science, Associate Professor, 2007-present, full-time
   - CSUN Computer Science, Assistant Professor, 2001-2007, full-time
   - CSUN Computer Science, Lecturer, 1999-2001, part-time
   - UCLA Computer Science, Research Assistant, 1998-2001, part-time
   - UCLA Computer Science, Teaching Assistant, 1994-1998, part-time

4. Non-academic experience
   - Hughes Research Laboratories, Programmer, 1997, part-time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - None

7. Honors and awards
   - Recognition as Faculty Advisor, CSUN chapter of ACM, 2007-2008

8. Service activities (within and outside of the institution)
   - Faculty Advisor, CSUN chapter of ACM, 2007-2008
   - Chair of Systems Group Assessment Committee, Computer Science Department, 2006-present
   - Chair of Part-time Faculty Review and Recruitment Committee, 2009-2011

9. Recent publications and presentations
   - M.D. Ercegovac and R. McIlhenny, *"Shared Implementation of Radix-10 and Radix-16 Square Root Algorithm with Limited Precision Primitives,"* 46th Asilomar Conference on Signals, Systems, and Computers, 2012
   - M.D. Ercegovac and R. McIlhenny, *"Shared Implementation of Radix-10 and Radix-16 Division Algorithm with Limited Precision Primitives,"* 45th Asilomar Conference on Signals, Systems, and Computers, 2011
   - M.D. Ercegovac and R. McIlhenny, *"Design and FPGA Implementation of Radix-10 Combined Division/Square Root Algorithm with Limited Precision Primitives,"* 44th Asilomar Conference on Signals, Systems, and Computers, 2010

- M.D. Ercegovac and R. McIlhenny, *"Design and FPGA of Radix-10 Algorithm for Square Root with Limited Precision Primitives,"* 43rd Asilomar Conference on Signals, Systems, and Computers, 2009
- R. McIlhenny and M.D. Ercegovac, *"On the Design of a Radix online floating-point multiplier,"* Proceedings of the SPIE Symposium on Advanced Signal Processing Algorithms, Architectures, and Implementations XIX, 2009
- M.D. Ercegovac and R. McIlhenny, *"Design and FPGA of Radix-10 Algorithm for Division with Limited Precision Primitives,"* 42nd Asilomar Conference on Signals, Systems, and Computers, 2008

10. Recent professional development activities
    - None

1. Name: **Gloria E. Melara**

2. Education
   - Ph.D. Information Technology, USC, 1994
   - M.S. Computer Science, Western Michigan University, 1983
   - M.A. Math Education, Western Michigan Univerisyt, 1980
   - B.S. Math, Universidad Nacional de El Salvador, 1978

3. Academic experience
   - CSUN, Computer Science, Full Professor, 2002, full-time
   - CSUN, Computer Science, Associate Professor, 1996, full-time tenured
   - CSUN, Computer Science, Assistant Professor, 1990, full-time probationary
   - CSUN, Computer Science, Lecturer, 1985, full-time

4. Non-academic experience
   - None

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM
   - SIGCSE

7. Honors and awards
   - Judge Julian Beck award. A joint project with Dr. Carol Shubin to implement "A Collaborative Interdisciplinary Environment to Host an Interactive Learning Research Experience", 2009
   - CO-IP and faculty mentor of US Department of Education HSI-STEM Grant, 2011-2016

8. Service activities (within and outside of the institution)
   - CSUN, Department Personnel Committee
   - CSUN, Department Assessment area coordinator
   - CSUN, College Personnel Committee
   - CSUN, Academic Technology Committee
   - CSUN, General Education Council
   - CSUN, Assessment liaison at university level
   - CSUN, International Liaison
   - CSUN, Adoc committees: Scholarship selection; faculty development grant committee
   - CSUN, Assistant Chair
   - Outside of CSUN: Reviewer panelist of SIGSE conference

- Advisory committee at Pierce Valley Community College, Los Angeles Community College; Los Angeles Valley Community College

9. Recent publications and presentations
   - None

10. Recent professional development activities
    - Attended -Intensive Scalable Computing in Education (DISC-E) a week workshop at the University of Washington, July 2008

1. Name: **Ani Nahapetian**

2. Education
   - Ph.D. Computer Science, UCLA, 2007
   - M.S. Computer Science, UCLA, 2004
   - B.S. Computer Science and Engineering, UCLA, 2002

3. Academic experience
   - CSUN Computer Science, Assistant Professor, 2011–present, full-time
   - CSUN Computer Science, Graduate Coordinator, 2011–present, full-time
   - UCLA Computer Science, Assistant Adjunct Professor, 2010-present
   - UCLA Computer Science, Visiting Assistant Professor, 2008-2010, part-time
   - CSU Dominguez Hills Computer Science, Assistant Professor, 2007-2008, on leave 2008-2010, full-time

4. Non-academic experience
   - Intel Corporation, Consultant, Expert Witness, 2011-2012, part-time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM
   - IEEE
   - CCSC

7. Honors and awards
   - Outstanding Engineering Achievement Merit Award, Engineers' Council, 2012
   - Best Paper Award, Conference on Mobile Computing, Applications, and Services (MobiCASE), 2009

8. Service activities (within and outside of the institution)
   - Member - CSUN Reading Initiative Committee, 2012-2013
   - Member - CSUN Computer Science Department Part-Time Lecturer Review Committee, 2011-2012
   - Associate Editor - ACM/Springer Wireless Networks, 2011- present
   - General co-Chair - IEEE International Conference Mobile Computing, Applications and Services (MobiCASE), 2011
   - Proceedings co-Chair - ACM SIGHIT International Health Informatics Symposium (IHI), 2012
   - Guest Editor - ACM Transactions on Embedded Computing Systems (TECS) Special Issue on Wireless Health Systems, 2009-2010

- Region Chair, Publicity Chair - Consortium of Computing Sciences in Colleges (CCSC) Southwestern Region, 2008-2011, 2012-present, respectively

9. Recent publications and presentations

- Addressing Responsiveness in Interactive, Pervasive Games. David Stepanyan, Ani Nahapetian. *IEEE ICME 2013 - International Workshop on Networking Issues in Multimedia Entertainment (NIME13)*, July 2013
- Gate Characterization Using Singular Value Decomposition: Foundations and Applications. Sheng Wei, Ani Nahapetian, Michael Nelson, Farinaz Koushanfar, Miodrag Potkonjak. *IEEE Transactions on Information Forensics and Security (TIFS)*, Vol. 7 No. 2, pp. 765-773, April 2012
- A Remote Patient Monitoring System for Congestive Heart Failure. Myung-kyung Suh, Chien-An Chen, Jonathan Woodbridge, Michael Kai Tu, Jung In Kim, Ani Nahapetian, Lorraine S. Evangelista, Majid Sarrafzadeh. *Springer Journal of Medical Systems (JOMS)*, Vol 35, No. 5, pp. 1165-1179, June 2011
- An Approximation Algorithm for Scheduling on Heterogeneous Reconfigurable Resources. Ani Nahapetian, Philip Brisk, Soheil Ghiasi, Majid Sarrafzadeh. *ACM Transactions on Embedded Computing (TECS), Vol. 9, No. 1*, October 2009
- General Methodology for Soft Error-Aware Power Optimization using Gate Sizing. Foad Dabiri, Ani Nahapetian, Tammara Massey, Miodrag Potkonjak, Majid Sarrafzadeh. *IEEE Transactions on CAD (TCAD), Vol. 27, No. 10*, October 2008
- Robust Passive Hardware Metering. Sheng Wei, Ani Nahapetian, Miodrag Potkonjak. *International Conference on Computer-Aided Design (ICCAD)*, November 2011
- Data Fusion for Movement Visualization in a Remote Health Monitoring System. Armen Babakanian, Ani Nahapetian. *IEEE International Conference on Mobile Computing, Applications, and Services (MobiCASE)*, October 2011
- Differential Public Physically Unclonable Functions: Architecture and Applications. Miodrag Potkonjak, Saro Meguerdichian, Ani Nahapetian, Sheng Wei. *Design Automation Conference (DAC)*, June 2011

10. Recent professional development activities
    - CCSC Southwestern Conference, CSU San Marcos, 2013
    - Intel Embedded Research and Education Summit, Chandler, AZ, 2013
    - California HSI Research Collaboration Conference, Anaheim, CA 2013
    - CCSC Southwestern Conference, Loyola Marymount University, 2011

1. Name: **John Noga**

2. Education
   - Ph.D. Mathematics, University of California Riverside, 1998
   - M.S. Applied Mathematics, University of California Riverside, 1992
   - B.S. Mathematics, Truman State University, 1991

3. Academic experience
   - CSUN Computer Science, Professor, 2012-Present, full-time
   - CSUN Computer Science, Associate Professor, 2007-2012, full-time
   - CSUN Computer Science, Assistant Professor, 2001- 2007, full-time
   - Technical University of Graz, Post-Doctoral Researcher, 1998-2000, full-time

4. Non-academic experience
   - None

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM (SIGACT)

7. Honors and awards
   - None

8. Service activities (within and outside of the institution)
   - Department Personnel Planning and Retention, 2011-2013
   - University Classroom Technology Committee
   - Supervision of Lab Tutors
   - Chair Search and Screen Committee (for Asst Prof position), 2011-2012
   - Chair Part-Time Lecturer Review Committee

9. Recent publications and presentations
   - Wolfgang Bein, Lawrence Larmore, John Noga, and Rudiger Reischuk, Knowledge State Algorithms, Algorithmica 60(3): 653-678 (2011)
   - Wolfgang Bein, Leah Epstein, Lawrence Larmore, and John Noga, Optimally Competitive List Batching, Theoretical Computer Science 410(38-40): 3631-3639 (2009)

10. Briefly list the most recent professional development activities
    - None

1. Name: **Son Pham**

2. Education
   - Ph.D. Mathematics and Statistics, University of Cincinnati-OH, 1978
   - M.A. Mathematics, University of Louisville, 1975
   - B.S. Mathematics, University of Saigon, 1973

4. Academic experience
   - CSUN, Computer Science, Full Professor, 1990-present, full-time
   - CSUN, Computer Science, Associate Professor, 1985-1989, full-time

5. Non-academic experience
   - None

6. Certifications or professional registrations
   - None

7. Current membership in professional organizations
   - None

8. Honors and awards
   - None

9. Service activities (within and outside of the institution)
   - CSUN, Member of Department Personnel Committee, 2012-2013
   - CSUN, Chair of Department Personnel Committee, 2009-2010
   - CSUN, Computer Science Thesis Advisor: Johnray Milano, 2013-present; Vu,Bao, 2013-present; Veronika Movagharianpour, 2012; Joel Hickman, 2009; Xiaoling Zhan, 2006

10. Recent publications and presentations
    - Dynamic External Merger Sort for Partial Group-by in Optimization, 9th Annual International Conference on Information Technology & Computer Science, 20-23 May 2013, Athens, Greece. Presenter:: Son Pham, Northridge, CA (US); Coauthor : Thu K Pham, 2013
    - Parsing Using An Object Oriented Database
      Patent Review Committee Complete (DN05-000041.00, CPI: 012200.00) Inventors: Son Pham, Northridge, CA (US); Thu K. Pham, Northridge, CA (US), 2012
    - Sorting of records with duplicate removal in a database system
      US Pat. 7370068 - Filed Sep 4, 2002 - Teradata US, Inc.
      Inventors: Son Pham, Northridge, CA (US); Thu K. Pham, Northridge, CA (US), 2008

11. Recent professional development activities
    - Learning Management System specific to the two courses: Computer Science 122 (Computer Architecture and Assembly) and Computer Science 440 (Database Design).
    - Web-base Algorithm Editor specific to Computer Science 110 (Introduction to Algorithms and Programming)

1. Name: **Diane Schwartz**

2. Education
   - Ph.D. Mathematics, UCLA, 1975
   - M.A. Mathematics, UCLA, 1971
   - B.A. Mathematics, UC Berkeley, 1966

3. Academic experience
   - CSUN Computer Science, Full Professor, 1983 – present, FERP
   - CSUN College of Engineering and Comp Science, Associate Dean, 1986-1989 and 2007-2009, full-time
   - CSUN College of Engineering and Comp Science, Interim Dean, 1992-1924 and 2001-2002, full-time
   - CSUN Computer Science, Department Chair, 1981- 1983 and 1994-1999, full-time
   - CSUN, Lecture/Associate Professor, 1979– 1983
   - USC, Lecturer, 1975-1977

4. Non-academic experience
   - Member of the Technical Staff, Jet Propulsion Laboratory, 1977 – 1979, full-time; 1979 – 1984, part-time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM

7. Honors and awards
   - San Fernando Valley Engineer's Council John Guarrera Distinguished Educator of the Year Award, 2007

8. Service activities (within and outside of the institution) within last 6 years
   - CSUN, WASC Accreditation Steering Committee and Co-Chair of a subcommittee which made major contributions to the campus WASC Self- Study Reports (Capacity and Preparedness Review and the Educational Effectiveness Review) , 2008-2011
   - CSUN, Department Assessment Activities 2007- present
   - CSUN, Department Assessment Coordinator 2011- present
   - CSUN, I wrote Criterion 2, 3, 4 sections of ABET report for 2013
   - CSUN, Faculty Governance: Faculty Senate, Senate Executive Committee, Educational Policies Committee , General Education Task Forces, 2009-present
   - CSUN, NSF Steps (Students Targeting Engineering and Physical Science) Grant, co-pi 2010-2011
   - SIGCSE reviewer 2007- present

9. Recent publications and presentations

- Wang, G., Schwartz, D, Lingard, R. Assessing Student Learning in Software Engineering. The Journal of Computing Science in Colleges. Volume 23, Number 6, 2008
- Presentation at WASC Academic Resource Conference 2010: “Using a Culture of Evidence to Advance Learning and Enhance Educational Effectiveness”, Schwartz, Diane, Cummins-Prager, Mary Ann, 2010

10. Recent professional development activities
    - Attended the SIGCSE Conference, March 2012
    - Attended the CAC ABET Accreditation Workshop, March 2012

1. Name: **Steven Stepanek**

2. Education
   - M.S. Computer Science, CSU, Northridge, 1980
   - B.A. Mathematics, CSU, Northridge, 1974

3. Academic experience
   - CSUN Faculty President, 2010-present
   - CSUN Computer Science, Department Chair, 1999-present, full-time
   - CSUN Computer Science, Full Professor, 1993-present, full-time
   - CSUN Computer Science, Associate Professor, 1985-1993, full-time
   - CSUN Computer Science, Assistant Professor, 1981-1985 (tenure in 1984), full-time
   - UCLA Extension, Computer Science Lecturer, 1984-1990, part-time
   - CSUN Computer Science, Lecturer, 1980-1981, part-time
   - CSUN Computer Science, Lecturer, 1978-1980, full-time
   - CSUN Computer Science, Lecturer, 1976-1978, part-time

4. Non-academic experience
   - Rand Corporation, Computer Consultant, 1984-1987, part-time
   - Control Data Corporation, Computer Consultant, 1980-1981, full-time
   - CSUN Computer Center, Staff, 1972-1978, full-time

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - ACM
   - IEEE Computer Society
   - USENIX

7. Honors and awards
   - Distinguished Engineering Educator Award From The Engineers' Council, 2011
   - CSU, Northridge Faculty Extraordinary Service Award, 2009
   - "Eminent Engineer" status in Tau Beta Pi, the Engineering Honor Society, 2006
   - Engineering Merit Award from The Engineers' Council, 2000

8. Service activities (within and outside of the institution)
   - CSUN, President's Extended Cabinet, 2012-present
   - CSU, Chair meetings of CSU Senate Chairs, 2011-present
   - CSU/CCC, C-ID Computer Science Course Reviewer, 2011-present
   - CSUN, Faculty President and Campus Senate Chair, 2010-present

- CSUN, Organizing Committee for “CSU Future” events, 2011-present
- CSUN, Search and Screen Committee for Campus President, 2011-2012
- CSUN, Senate Personnel Planning and Review Committee, 2010-present
- CSU, Vice Chair of CSU Admissions Advisory Council, 2008-present
- CSUN, Campus Technology Infrastructure and Services Planning Committee, 2006-present
- CSUN, Provost’s Advisory Committee on Academic Technologies, 2006-present
- CSU, Academic Senate, 2005-present
- CSUN, General Education Reform Committee, 2004-2005
- CSUN, Campus WASC Accreditation Steering Committee, Co-Chair of Technology Subcommittee, 1998-2000

9. Recent publications and presentations
    - Presentation on Cloud Computing Technologies, CSU Academic Resource Conference, San Diego, March 2013
    - Annual presentation on “Policies Dept Chairs Should Know”, CSUN Dept Chairs and Administrators Retreat, August 2010-present

10. Recent professional development activities
    - Annual CSUN Department Chairs and Administrators Retreats, August 1999-present
    - CSUN Professional Development Seminars sponsored by the Provost’s Office approximately 4 times per year, 2008-present

1. Name: **Taehyung (George) Wang**

2. Education
   - Ph.D. Electrical Engineering and Computer Science, UC, Irvine, 1998
   - M.S. Computer Science, Western Illinois University, 1994
   - B.S. Electrical and Engineering and Computer Science, Seoul National University, Seoul, Korea, 1985

3. Academic experience
   - CSUN Computer Science, Associate Professor, 2009 – Present
   - CSUN Computer Science, Assistant Professor, 2003 – 2009
   - University of California, Irvine, EECS Department, Researcher, 2000 – 2003
   - University of California, Irvine, Center of Biomedical Engineering, Adjunct Researcher, 2001 – 2002
   - University of California, Irvine, EECS Department, Postdoctoral Researcher, 1999 – 2000

4. Non-academic experience
   - LG Information and Telecommunication, Korea Software Engr, 1985 – 1991

5. Certifications or professional registrations
   - None

6. Current membership in professional organizations
   - IEEE-CS; ASEE; KSEA (Korean-American Scientists and Engineers Association); KOCSEA (Korean Computer Scientists and Engineers Association in America)

7. Honors and awards
   - Principle Investigator, "The Next Generation Carelink Towards a Diabetes Information Framework," Medtronic MiniMed Grant, 2011-2012, $ 34,951
   - Distinguished Engineering Achievement and Engineering Educator Award, The San Fernando Valley Engineers' Council, 2010
   - Principle Investigator, "Knowledge-based Testing and Evaluation Environment for Insulin Pump Software," Medtronic MiniMed Grant, 2005-2007, $33,539
   - Principle Investigator, "Bioinformatics Framework: Semantic Integration of Heterogeneous and Distributed Bioinformatics Databases and Agent-based Information Retrieval," CSUN Competition for Research, Scholarship and Creative Activity, Jan. 2005 – May 2005, $4,363

8. Service activities (within and outside of the institution)

- CSUN: Department Program Improvement Committee, 2011-present; Research and Grants Committee, 2011-Present; Department ABET Assessment Committee, 2004-present;
- Finance Chair for the IEEE International Symposium on Multimedia, Irvine, California, Dec 10–12, 2012; Dana Point, California, Dec 5–7, 2011.
- Finance Chair for the IEEE International Conference on Semantic Computing, Palermo, Italy, Sept 19–21, 2012; Stanford, California, Sept 19–21, 2011.

9. Recent publications and presentations
   - Chanmin Park, Seunghyun Kim and Taehyung (George) Wang, "Multimedia Copyright Protection on the Web - Issues and Suggestions," *IEEE International Symposium on Multimedia (ISM2012)*, Irvine, California, December, 2012.
   - Min Kyo Chung, Taehyung (George) Wang and Phillip Sheu, "Folksonomic Approach to Video Summarization," *Online Information Review*, Emerald, vol. 35. no. 4. 2011.
   - Wooju Kim , June S. Hong, Taehyung (George) Wang, Myungjin Lee and Hak-Jin Kim, "Mathematical Constraint Knowledge into the Semantic Web by a Semantic Web Constraint Language," a chapter of the book entitled *Introduction to the Web Semantic: Concepts, Technologies and Applications*, iConcept Press, 2011.
   - Myungjin Lee, Wooju Kim, and Taehyung (George) Wang, "An Explorative Association-Based Search for the Semantic Web," *IEEE International Conference on Semantic Computing (ICSC 2010)*, Pittsburgh, Pennsylvania, September 22–24, 2010.
   - ChengZhi Xu, Peng Liang, Taehyung (George) Wang, Qi Wang, and Phillip C-Y, Sheu "Semantic Web Services Annotation and Composition based on ER Model," *IEEE International Workshop on Ubiquitous and Mobile Computing (UMC 2010)*, Newport Beach, California, June 7–9, 2010.
   - Christopher N. Gutierrez, Gautam Kakani, Ramesh C. Verma, Taehyung (George) Wang, "Digital Watermarking of Medical Images for Mobile Devices," *IEEE International Workshop on Ubiquitous and Mobile Computing (UMC 2010)*, Newport Beach, California, June 7–9, 2010.
   - Seung-Hyun Kim, Craig Dunham, Suryo Muljono, Albert Lee, and Taehyung (George) Wang, "Discovery of Association Rules in National Violent Death Data Using Optimization of Number of Attributes," *World Congress on Computer Science and Information Engineering (CSIE 2009)*, Los Angeles/Anaheim, California, March 31–April 2, 2009.
   - Jordan Yelloz and Taehyung (George) Wang, "Optimization and Load Balancing of the Semantic Tagging and Searching System," *IEEE International Workshop on Semantic Computing and Applications (IWSCA 2008)*, Incheon, Korea, July 10–11, 2008.
   - Taehyung (George) Wang, Diane Schwartz, and Robert Lingard, "Assessing Student Learning in Software Engineering," *The Journal of Computing Science in Colleges*, vol. 23, no. 6, June, 2008, pp. 239 – 248.

10. Recent professional development activities
    - None

1. Name: **Jeffrey Wiegley**

2. Education
   - Ph.D. Computer Science, University of Southern California, 1998
   - B.S. Computer Science, State University of New York at Buffalo, 1991

3. Academic experience
   - CSUN Computer Science, full-time

Minutes from Meetings Document Faculty Consultation on Program Review

Computer Science Department Minutes - October 12, 2012

Present: Shan Barkataki, Mike Barnes, Rick Covington, Peter Gabrovsky, Adam Kaplan, Robert Lingard, Richard Lorentz, Robert McIlhenny, Gloria Melara, Ani Nahapetian, John Noga, Pham Son, Diane Schwartz, Steven Stepanek, George Wang, Bahram Zartoshty

Absent/Excused: Steven Fitzgerald, Jeff wiegley

Recorder: Bahram Zartoshty

1. **Graduate Program Self-Study Report Planning**

- Led by Ani Nahapetian, the Department had a discussion on the graduate program self-study planning. The following weaknesses were discussed:
    - Students lacking writing and communication skills necessary to successfully complete the requirements for MS in Computer Science.
    - Majority of students disappear from the program due to the fact they can not finish their thesis.
    - Lack of job opportunities in the department (lab assistant, teaching assistant, etc.).
    - University GRE and TOEFL requirements has limited number of international students admitted to Computer Science graduate program.
    - Some courses are not offered as frequently as needed (low enrollment).

    Faculty suggestions

    - Students should have the following choices to complete Computer Science graduate program:
        - Thesis
        - Comprehensive exit exam
        - Project based exit course (3-6 units)
        - Project/exam
    - Graduate Program Coordinator should individually evaluate student's admission test results. Each student is admitted to program based on the Graduate Coordinator recommendation.

2. **Graduate Program Update**

Bob Lingard made the following announcements:

- Deadline to submit objectives for 600 level courses: October 12, 2012
- Graduate Studies has notified that "12 unit" rule is no longer in effect.

Current issues in Graduate program

- **University policy:** Graduate students should form their committee as soon as they are admitted to the program.
- **Department policy (catalog):** Graduate students should form their committee as soon as they become classified (enroll in Comp 696).

Discussion: What is department's position? Should we continue what is stated in catalog or follow university policy. Discussion to be postponed to future department meetings.

Catalog issues:

- All editorial issues should be reported to Steven Stepanek. They can easily be fixed
- Handout given by Bob Lingard indicates there are several discrepancies with regard to "Upper Division Writing Proficiency Exam" prerequisite. Three different phrases used are as follows:
    - Attempted Upper Division Writing Proficiency Exam
    - Passing score on the Upper Division Writing Proficiency Exam
    - Upper Division Writing Proficiency Exam
- Department approved to remove "Attempted Upper Division Writing Proficiency Exam" as prerequisite for all 400 level courses except for Comp450 and Comp 490.
- Department approved to make "Upper Division Writing Proficiency Exam" as the prerequisite for Comp 450 and Comp 490.
- Department approved for all other courses the prerequisite should be "Upper Division Writing Proficiency Exam".

Proposed Changes to MS in Software Engineering Prerequisites

- Department approved the following:
    - Courses eliminated: Comp 230, Comp310, Math 482
    - Courses added: Math 340 or Math 341

Computer Science Department Meeting Minutes -March 9, 2012

Attendees: Shan Barkataki, Mike Barnes, Rick Covington, Peter Gabrovsky, Bob Lingard, Robert McIlhenny, Gloria Melara, Ani Nahapetian, John Noga, Son Pham, Diane Schwartz, Steven Stepanek (Chair), George Wang, Jeff Weigley, Bahram Zartoshty

Recorder: Ani Nahapetian

Graduate Program: - Robert Lingard gave a listing of all the graduate course listings missing objectives and the faculty members tasked with completing them. Comp 667 is still in the catalogue, even though it should have been archived.

Although there is an understanding that undergraduate students should not take 600 level classes, there is no place where this is explicitly listed. There is only a mention that 600 level courses do not contribute to an undergraduate degree. Shan Barkataki moved to allow undergraduates to enroll in 600 level courses in their senior year as long as they are properly advised. The motion passed. Steven Stepanek will continue to contact undergraduate students who enroll in 600 level courses to make sure they are aware of the type of course they are enrolled in.

The Program Review: Self-Study Guidelines were provided to all faculty members. The last graduate program review was eight years ago, and so the next program review is now being considered. A motion was made and passed to have the graduate coordinator, the graduate assessment coordinator, and to a lesser role the department chair be involved in the report preparation. The current graduate coordinator and the next graduate coordinator will work together in Fall 2012 to ease the transition, and both will contribute to the report preparation.

Computer Science Department Meeting Minutes - December 9, 2011

Present: Jack Alanen, Shan Barkataki, Mike Barnes, Rick Covington, Peter Gabrovsky, Bob Lingard, Rich Lorentz, Bob McIlhenny, Gloria Melara, John Noga, Son Pham, Diane Schwartz, Steven Stepanek, Ginter Trybus, Jeff Wiegley

Excused: Steven Fitzgerald, Ani Nahapetian, George Wang, Bahram Zartoshty

Recorder: Bob Lingard

Comp 696/698 Project/Thesis Committee Report: Shan Barkataki distributed a report from the committee and a long discussion followed. The report highlighted several issues with the current thesis/project process and procedures. First it was noted that students seem ill prepared when it comes to starting a thesis or project. As a result one of the main recommendations was that the Graduate Coordinator conduct a class as a part of Comp 696C to make sure students get started correctly. Other problems noted were that students frequently do no accomplish much in the first semester (Comp 696C) and that many students have difficulty in completing the thesis or project or take too long to finish. Recommendations for these problems included identifying specific deliverables that students must produce at specific times during the thesis/project timeframe. Many other problems were identified including the wide variance in content and quality of the student work, the large amount of work that is often required by the committee chair, deadlines being ignored by the students and not enforced by the department, students not following correct procedures for distributing their theses/projects to committee members or scheduling their defenses, and a general lack of uniformity in the faculty practices with regard to thesis/project work. A separate but related issue is the fact that the use of a thesis/project is the only culminating experience avenue offered to our students. The committee discussed briefly that adding additional options might alleviate some of the current problems but recommended that this be handled by a separate committee. However, a motion was made, seconded and passed to form one new committee to make recommendations for addressing both problems, i.e., difficulties with the current thesis/project option and investigate the possibility of adding an additional option or options for the culminating experience. The following faculty members were elected to serve on this committee: Barkataki (chair), Lingard, Lorentz, Pham, Wiegley.

Data from Institutional Research with Analysis

We have obtained data from CSUN's Institutional Research (IR) to study various demographic trends in the Department's graduate programs. This data, presented in Tables 4, 5, 6, 7, 8, and 9, with the respective analysis is provided in this section. Note that the data presented in this section groups together both of Department's graduate programs, Computer Science and the Software Engineering.

Table 4 shows the average entering GPAs of students in the Computer Science graduate program for nine separate years. The average GPA is above a 3.4 on average, implying that most students who complete the program are in good standing at the completion of the degree.

Numbers	AY 1990/91	AY 1995/96	AY 2000/01	AY 2005/06	AY 2007/08	AY 2008/09	AY 2009/10	AY 2010/11	AY 2011/12
GPA	3.42	3.51	3.51	3.48	3.64	3.41	3.44	3.53	3.38

Table 4: Computer Science Department Master's Programs Average CSUN GPA at Exit

Table 5 shows the average entering GPAs of students in the Computer Science graduate program for nine separate years. The average GPA of the students upon entry into the Computer Science Department's Master's programs in high and steadily rising. Additionally, the average GPA is higher than the minimum GPA of 3.0 required by the Department by about 0.4 points, signaling that the department's standards are high and that we attract strong students. Moreover, it hints to the fact that the GRE score is more aggressive selecting students than the GPA requirement directly.

	Fall 1993	Fall 1995	Fall 2000	Fall 2005	Fall 2008	Fall 2009	Fall 2010	Fall 2011	Fall 2012
Average GPA	3.02	3.49	3.49	3.51	3.42	3.36	3.39	3.46	3.48

Table 5: Computer Science Department Master's Programs Average CSUN GPA at entry

There has been a significant number of international applicants to the graduate programs, however, their acceptance rate and attendance rate is much lower than their application rates. Table 6 shows that the number of international students has ranged from 20-30% in the past 5 years, although their application rates are well over 50%.

The number of Hispanic students in the graduate programs has consistently been around 10%, as shown in Table 6. This is a significantly less than the undergraduate Computer Science program, which has seen a rise in the Hispanic student population to around 25%, as shown in Table 7.

	Fall 1993	Fall 1995	Fall 2000	Fall 2005	Fall 2008	Fall 2009	Fall 2010	Fall 2011	Fall 2012
Traditionally Underserved	7.4%	7.8%	3.6%	6.5%	14.9%	9.6%	12.0%	13.1%	13.9%
American Indian/Alaskan Native	0.8%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Native Hawaiian/Pacific Islander	0.0%	0.0%	0.0%	0.0%	1.5%	2.7%	4.0%	1.6%	0.0%
African American	2.5%	1.0%	2.4%	0.0%	1.5%	1.4%	2.7%	1.6%	2.8%
Latina/o	4.1%	5.9%	1.2%	6.5%	11.9%	5.5%	5.3%	9.8%	11.1%
Asian	23.1%	29.4%	10.7%	14.5%	20.9%	16.4%	12.0%	11.5%	15.3%
White	48.8%	33.3%	31.0%	32.3%	25.4%	39.7%	38.7%	47.5%	41.7%
Multi-Race/Other	0.8%	3.9%	7.1%	3.2%	1.5%	1.4%	0.0%	1.6%	2.8%
Unknown	11.6%	11.8%	8.3%	16.1%	13.4%	8.2%	6.7%	4.9%	6.9%
International	8.3%	13.7%	39.3%	27.4%	23.9%	24.7%	30.7%	21.3%	19.4%
Total	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

Table 6. Computer Science Department Master's Programs Headcount Percentages by Ethnicity

	Fall 1993	Fall 1995	Fall 2000	Fall 2005	Fall 2008	Fall 2009	Fall 2010	Fall 2011	Fall 2012
Traditionally Underserved	20.0%	23.7%	20.7%	21.1%	22.0%	20.7%	24.9%	30.2%	30.3%
American Indian/Alaskan Native	0.0%	0.2%	0.8%	0.4%	0.6%	0.3%	0.3%	0.0%	0.6%
Native Hawaiian/Pacific Islander	0.6%	0.4%	0.3%	0.8%	0.0%	0.0%	0.0%	0.2%	0.2%
African American	7.0%	7.8%	5.7%	5.9%	5.6%	3.3%	4.7%	4.5%	3.1%
Latina/o	12.4%	15.3%	13.9%	13.9%	15.8%	17.1%	20.0%	25.5%	26.4%
Asian	25.3%	30.5%	28.1%	21.1%	19.2%	16.8%	17.7%	17.7%	17.6%
White	39.5%	30.1%	30.4%	29.1%	34.6%	35.9%	33.8%	31.3%	31.2%
Multi-Race/Other	3.6%	3.9%	5.5%	4.0%	0.6%	1.1%	2.6%	2.8%	4.0%
Unknown	7.4%	6.7%	8.2%	13.1%	11.8%	12.5%	10.4%	6.0%	6.7%
International	4.2%	5.1%	7.0%	11.8%	11.8%	13.0%	10.6%	11.9%	10.2%
Total	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

Table 7. Computer Science Undergraduate Headcount Percentages by Ethnicity

Tables 8 and 9 provide the headcounts and the percentages, respectively, of students enrolled in the Computer Science Master's Programs with full-time and part-time status. The numbers indicate that most students are part-time, probably due to constraints imposed by full-time employment outside the University. However, the number of full-time students is growing over the recent years.

Numbers	Fall 1993	Fall 1995	Fall 2000	Fall 2005	Fall 2008	Fall 2009	Fall 2010	Fall 2011	Fall 2012
Full-Time*	22	23	36	20	22	28	25	23	25
Part-Time*	99	79	48	42	45	45	50	38	47
Total	121	102	84	62	67	73	75	61	72

*Full-Time Undergraduates Attempting 12 or more hours

*Full-Time Graduates Attempting 9 or more hours

Table 8. Computer Science Department Master's Programs Headcount by Attendance Status

Percent	Fall 1993	Fall 1995	Fall 2000	Fall 2005	Fall 2008	Fall 2009	Fall 2010	Fall 2011	Fall 2012
Full-Time*	18.2%	22.5%	42.9%	32.3%	32.8%	38.4%	33.3%	37.7%	34.7%
Part-Time*	81.8%	77.5%	57.1%	67.7%	67.2%	61.6%	66.7%	62.3%	65.3%
Total	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

*Full-Time Undergraduates Attempting 12 or more hours

*Full-Time Graduates Attempting 9 or more hours

Table 9. Computer Science Department Master's Programs Headcount Percentages by Attendance Status