

School of Computing Graduate Handbook

**University of Utah
School of Computing
50 S Central Campus Dr RM 3190
Salt Lake City, UT 84112-9205**

**(801) 581-8224 (voice)
(801) 581-5843 (fax)
info@cs.utah.edu
http://www.cs.utah.edu**

2002-2003

In 1965, the Department of Computer Science was founded. Its name was changed to the School of Computing on July 1, 2000, to reflect the broadening of academic pursuits already underway within the faculty, and to encourage the creation of multi-disciplinary programs with other university departments. The degree programs remain as degrees in Computer Science.

The School of Computing offers highly-regarded programs at both the undergraduate and graduate levels. School faculty and students have done pioneering work in interactive graphics, stack machine and dataflow architectures, digital recording, graphical user interfaces, three-dimensional rendering, asynchronous circuits, video games, computer algebra, and computer animation. Faculty and alumni have also founded a number of well-known companies, including Adobe Systems, Ashlar, Atari, Cirrus Logic, Evans & Sutherland, Myricom, Netscape, Pixar, Pixel-Planes, Silicon Graphics, and WordPerfect.

Graduate students immerse themselves in the research activities of the School, which can be approximately grouped into 7 broad areas. Many individual research groups do work spanning multiple areas.

1. *Graphics and visualization*: modeling, CAD/CAM, rendering, and scientific visualization.
2. *Systems and programming languages*: operating systems, parallel distributed systems, programming languages, compilers, security, networks, software engineering.
3. *Immersive environments*: novel systems for building immersive environments allowing manipulation of virtual objects and locomotion through virtual worlds.
4. *Architecture, VLSI, and verification methods*: Innovative memory and communication architectures, asynchronous circuits and systems, formal verification of computing systems.
5. *Scientific computation*: methods for solving and visualizing large-scale scientific problems.
6. *Artificial intelligence*: natural language processing, robotics, and computer vision.
7. *Educational software environments*: applications of computing, communications, and connectivity to education.

These research activities are funded from a variety of federal, state, and industrial sources, including: the National Science Foundation (NSF), the Defense Advanced Research Projects Agency, the Department of Energy, the Office of Naval Research, and the National Institutes of Health; the Utah State Centers of Excellence Program; and Adobe, Cisco, Evans and Sutherland, Hewlett Packard, Microsoft, Nortel, Novell, and Silicon Graphics. Among the highlights:

- The School is a partner in an NSF Science and Technology Center for Computer Graphics and Visualization along with Brown, Caltech, Cornell, and the University of North Carolina.
- An NSF Research Infrastructure award supports research activities requiring high-bandwidth, low-latency machine-to-machine communications.
- It is a partner in the DOE Advanced Visualization Technology Center along with Los Alamos National Laboratory and Argonne National Laboratory.
- It is a key participant in the University's \$20 million Accelerated Strategic Computing Initiative grant from the Department of Energy.
- It has an NIH Center for Bioelectric Field Modeling, Simulation, and Visualization.

Reflecting the broadened mandate that went into the creation of the School of Computing, the Scientific Computing and Imaging (SCI) Institute was recently formed within the School. Also, the School recently received the John E. and Marva M. Warnock Presidential Endowed Chair for Faculty Innovation, to support scholarship and creativity in the early career of an outstanding young faculty member. John Warnock is co-founder and president of Adobe.

Graduate students have access to hundreds of Unix and Windows NT workstations and to the more specialized equipment that resides in the various research laboratories. This equipment includes a 96 CPU SGI Origin 2000 with 8 Infinite Reality Engines; a 200-node network testbed and emulation facility; SGI ONYX2, Power Challenge, Power Onyx, and Origin 200 computers; robot arms, mobile robots, and image digitization and display systems; a variety of visual, haptic, and locomotory virtual environment interfaces; a professional-quality video editing and teleconferencing facility; advanced graphics display workstations equipped with special-purpose graphics hardware; and a collection of numerically controlled equipment used to produce physical prototypes of computer-generated designs.

The University of Utah is committed to policies of equal opportunity, affirmative action, and nondiscrimination. The University seeks to provide equal access to its programs, services, and activities for people with disabilities. Reasonable prior notice is needed to arrange accommodations.

(This handbook is available online at <http://www.cs.utah.edu/dept/handbooks>.)

Faculty

Professors	Elaine Cohen, Ph.D. Ganesh Gopalakrishnan, Ph.D. Lee A. Hollaar, Ph.D. Christopher R. Johnson, Ph.D. Gary E. Lindstrom, Ph.D. Kris Sikorski, Ph.D. Frank Stenger, Ph.D.	Alan L. Davis, Ph.D. Thomas C. Henderson, Ph.D. John M. Hollerbach, Ph.D. Robert Kessler, Ph.D. Richard F. Riesenfeld, Ph.D. Kent F. Smith, Ph.D. William B. Thompson, Ph.D.
Clinical Professors	David Hanscom, Ph.D.	Joseph L. Zachary, Ph.D.
Research Professor	Stephen Jacobsen, Ph.D.	
Emeritus Professors	Robert R. Johnson, Ph.D.	Kent Smith, Ph.D.
Associate Professors	Erik Brunvand, Ph.D. Charles Hansen, Ph.D. Peter Shirley, Ph.D.	John Carter, Ph.D. Ellen M. Riloff, Ph.D.
Clinical Associate Professor	Art Lee, Ph.D.	
Research Associate Professors	Sam Drake, Sc.D.	Jay Lepreau
Assistant Professors	Matthew Flatt, Ph.D. Mike Kirby, Ph.D. Konrad Slind, Ph.D. Ross Whitaker, Ph.D.	Wilson Hsieh, Ph.D. Emil Praun, Ph.D. Cynthia Thompson, Ph.D.
Research Assistant Professors	Yarden Livnat, Ph.D.	Steven Parker, Ph.D.
Adjunct Professor	Martin Griss, Ph.D.	
Adjunct Associate Professors	Robert McDermott, Ph.D.	Chris Myers, Ph.D.
Adjunct Assistant Professor	Sally McKee, Ph.D.	

Administration

Thomas C. Henderson	Director	581-3601
Charles Hansen	Associate Director	581-3154
Dave Hanscom	Director, Undergraduate Studies	581-7023
Ross Whitaker	Director, Graduate Studies	587-9549
Kris Sikorski	Director, Computational Engineering and Science	581-8579
Neil Cotter	Director, Computer Engineering	581-8566
Joseph L. Zachary	Director, Educational Programs	581-7079

Administrative Staff

Erin Davies	Administrative Assistant	581-8226
Ann Torrence	Program Development Director	581-7631
Phil Willden	Administrative Manager	581-3950
Sara Hanney	Administrative Secretary	581-8226

Academic Advising

Marilyn Gorder	Graduate Coordinator	585-3551
Sandy Hiskey	Undergraduate Academic Counselor	581-8224
TBN	Administrative Secretary	581-8224

Technical Staff and Services

Corey Hatch	Digital Systems Laboratory Manager	581-7845
Monica Heaton	Administrative Officer	581-9371
Chad Lake	Computing Facility Director	585-7614
TBN	NT Lab Manager	585-9047

A complete staff listing can be found at <http://www.cs.utah.edu/people.html>.

Contents

Faculty	iii
Administration and Staff	v
Contents	viii
1 M.S. and Ph.D. Admissions	1
1.1 Required Application Materials	1
1.2 Application Deadline Dates	2
1.3 Addresses	2
1.4 If You Are Accepted	2
1.5 Frequently Asked Questions	3
1.6 Transfers Within the Graduate Program	4
1.6.1 Transfer into the Ph.D. Program	4
1.6.2 Transfer into the M.S. Program	4
2 Information for Graduate Students	5
2.1 The Graduate Studies Committee	5
2.2 The Graduate Admissions Committee	6
2.3 The Graduate Coordinator	6
2.4 Financial Assistance For M.S. and Ph.D. Students	6
2.5 Transfer Credit	8
2.5.1 All Students	8
2.5.2 Ph.D. Students	8
2.6 Leave of Absence Policy	8
2.7 GPA Requirements	9

2.8	Course Load	9
2.9	The M.S. Degree in Computer Science	9
2.10	The Ph.D. Degree in Computer Science	11
2.11	M.S. Thesis and Ph.D. Dissertation Requirements	13
2.12	Thesis or Dissertation Copyright Policy	14
2.13	The M. Phil. Degree	14
3	Computing Facilities	15
4	Computer Science Courses	17
5	School of Computing Faculty and Their Research Interests	29
6	Current Funded Research	47

1

M.S. and Ph.D. Admissions

The School of Computing conducts an active and well-funded research program, which allows it to provide high-quality graduate education and research experience to a select group of creative and highly motivated graduate students. The School of Computing offers the M.S. in Computer Science and the Ph.D. in Computer Science degree programs. Most graduate students are supported financially throughout their graduate career via a combination of teaching assistantships (TAs) and research assistantships (RAs). Our admissions standards are high and competition for the limited number of positions in the program is rigorous. Admission is based on an evaluation of both an applicant's *academic profile* and *research potential*. We especially encourage applications from underrepresented minorities and women.

1.1 Required Application Materials

The following materials are required of **ALL** applicants to either the M.S. or Ph.D. programs, including those wishing to transfer into the program from another department at the University of Utah.

1. **Application Form:** Complete and return the School of Computing Application for Admission to Graduate School. It is strongly encouraged that applicants use the on-line web form at <http://www.cs.utah.edu/dept/admissions/index.html>. Otherwise, you should download a hardcopy of the form from the same site; no forms will be mailed.

There is no application fee for applying to the School of Computing. However, once a student is accepted into our graduate program and decides to attend, the student is then required to submit another application form to the University of Utah. The application fee for the University is \$40 for domestic students and \$60 for international students.

2. **Grade Reports and Transcripts:** In accordance with the instructions on the application form, arrange to have two copies of your transcripts sent to the School of Computing directly from the issuing schools.
3. **GRE Aptitude and Advanced Test:** Applicants to the School of Computing at the University of Utah must take the GRE General Examination. Those qualified to do so are also encouraged to take an appropriate Subject Exam. Examination scores must be sent to the School of Computing by ETS - photocopies of scores will not be accepted. Students should take the GRE exam sufficiently early to allow their scores to arrive by the application deadline. Applicants who hold an advanced degree in computer science or another technical field may petition to omit GRE scores.

4. **Letters of Recommendation:** Please obtain hardcopies of the forms for letters of recommendation at the same web site as above. Arrange to have three letters of recommendation sent directly to the School of Computing *Graduate Coordinator*. We strongly prefer that all recommendations be from current or former professors who have a knowledge of the applicant's abilities through classwork, independent study, and/or research. However, a letter from a professional supervisor is acceptable if the supervisor is from a computer-related field and is qualified to judge the academic and technical capabilities, character, and ability of the applicant to perform research. *It is helpful if letters for non-native English speakers address the applicant's English skills.*
5. **Personal Letter:** Send a one- to two-page letter to the School of Computing *Graduate Coordinator*, describing in depth your background, interests, and in particular, your reasons for wanting to pursue graduate studies in computer science at the University of Utah.
6. **TOEFL Scores:** Applicants whose native language is not English must take the TOEFL (Test of English as a Foreign Language) and have the score reported to the School of Computing.

Note for foreign applicants: Letters of recommendation and your own personal letter must be written in English.

1.2 Application Deadline Dates

Applicants are normally evaluated for admission effective Fall Semester. Such applications must be received at the University of Utah by January 15 of the academic year prior to the desired start of studies. Exceptionally well qualified students may be admitted starting Spring Semester with an application deadline of October 1. Applications for entry effective Summer Term are not accepted.

1.3 Addresses

Applicants should have *all* application materials (application form, transcripts, GRE scores, letters of recommendation, TOEFL scores (if applicable), and personal letter) sent directly to the School of Computing at the address below. Questions regarding the status of an application or regarding the computer science graduate program in general should be sent to the Graduate Coordinator.

Graduate Coordinator
University of Utah
School of Computing
50 S. Central Campus Drive, RM 3190
Salt Lake City, UT 84112-9205 USA
grad-coordinator@cs.utah.edu
(801) 581-8224

1.4 If You Are Accepted

Once you have received your letter of acceptance from the School of Computing and decide to attend, be sure to notify the School of Computing in writing regarding your intention to matriculate. You will then be required to submit another application form to the University of Utah. The application fee for the University is \$40 for domestic students and \$60 for international students. If you are accepted with financial aid, obtain a TA/RA Request form from the Graduate Coordinator and return it promptly.

International Teaching Assistants (TAs who are not native speakers of English) must take either a Test of Spoken English (TSE) or SPEAK test before beginning their first TA assignment. Depending on exam score, remedial instruction at the

University's English Language Institute may be required. International Teaching Assistants are required to participate in the ITA Workshop prior to their first semester. This is typically held during the two weeks prior to start of the semester.

1.5 Frequently Asked Questions

What kind of academic background is required?

Neither of the graduate programs (M.S. or Ph.D) is an entry-level degree. Although it is not necessary that you have a B.S. in Computer Science, both degree programs assume a background in the core areas of computer science. If you don't have this background, you'll have to make it up once you start your graduate studies by taking appropriate courses. Please note though that undergraduate courses are not supported by the tuition waiver program for graduate students. See the next Chapter about the tuition waiver program.

Is there a GPA cutoff?

Yes, a minimum GPA of 3.0 in undergraduate work is required. Most students accepted into the graduate program have GPAs well above that level.

Is the GRE required?

Applicants are *required* to take the GRE General exam. Applicants who demonstrate greater depth of knowledge through GRE Subject exam will be given favorable consideration. The School of Computing prefers the GRE Computer Science Subject exam, but will consider GRE Subject exams in other technical specialities. Applicants who hold an advanced degree in computer science or another technical field may petition to omit GRE scores.

Is there a TOEFL cutoff?

Yes, the minimum acceptable score is 260 (620 in the old, paper-based testing scheme). In addition, any non-native English speaker who receives support as a teaching assistant must attend and pass the appropriate University programs in spoken English.

What is the application fee? Can it be waived?

There is no application fee for the School of Computing. However, students who are accepted into the School's graduate program and decide to attend, must then submit an application form to the University of Utah. The University application fee is \$40 for domestic applicants and \$60 for foreign applications. It cannot be waived under any circumstances.

Is there anything else involved in the admission process besides sending my application to the School of Computing?

Yes, you must supply the School with three letters of recommendation, academic transcripts, a personal statement, official GRE scores, and TOEFL scores (if applicable). Your application cannot be processed by the School's Graduate Studies Committee until it is complete.

Can I get more information on the School or copies of the admissions forms electronically?

Yes. You can get copies of this handbook and all of the admissions forms as well as detailed information about individual researchers' interests, ongoing research projects, and recent technical papers via the School's World Wide Web (WWW) server, located at <http://www.cs.utah.edu>. No admissions forms will be mailed; you must obtain them electronically.

Do I have to contact a faculty member to support my admission?

No. Students are accepted to the School, rather than to a specific research group, by a general admissions committee. Of course, you should feel free to contact a research group if you have questions about their research. Once you begin your studies here, you are encouraged to join a particular group whose research interests you at any time. Sending multiple emails to faculty asking for support or looking to join a group before you are admitted does not advance the process.

What are you really looking for in an applicant?

We are looking for applicants with strong academic backgrounds who have demonstrated a potential to perform creative and innovative research in the research areas represented by the School of Computing.

Who should I contact if I have any questions?

Marilyn Gorder (gorder@cs.utah.edu), the Graduate Coordinator.

1.6 Transfers Within the Graduate Program

The following section details School policy and procedures for currently matriculated School of Computing graduate students who wish to change degree programs (M.S. to Ph.D. or vice versa), and students who are about to complete a degree program and wish to continue in another. All such applications will be reviewed by the Graduate Studies Committee to ensure standards consistent with those applied to outside applicants. The same admissions deadlines apply as for outside applicants (i.e., a student wishing to transfer from the M.S. program for the Fall Semester must submit their application by January 15).

1.6.1 Transfer into the Ph.D. Program

A student wishing to transfer to the Ph.D. program must submit the following material to the Graduate Studies Committee:

1. A “goal letter” explaining the student’s motives and describing the intended research area.
2. Letters from three current School faculty members supporting the application. In the event that the applicant’s supervisory committee has been formed, then at least one of these letters must be from the chair of that committee.
3. A University of Utah transcript.

Applications will not be reviewed unless at the time of application the applicant has completed at least two academic semesters of graduate study. Summer term does not count in this tally.

1.6.2 Transfer into the M.S. Program

Students currently enrolled in the Ph.D. program who wish to transfer to the M.S. should make their request in writing to the Director of Graduate Studies. Supporting letters from three faculty members expressing willingness to serve on the student’s M.S. Supervisory Committee must also be provided.

2

Information for Graduate Students

The following summarizes the formal requirements and procedures of the M.S. in Computer Science and Ph.D. in Computer Science degree programs. Other pertinent information can be obtained from the bulletin of the Graduate School of the University of Utah.

Many aspects apply to both the M.S. and Ph.D. degree programs in Computer Science. The M.S. degree requires a thesis, and the Ph.D. degree a dissertation, to be completed and successfully defended before the degree can be awarded. (In the discussion that follows, the term “thesis” is sometimes used to refer to both.) Information that applies specifically to M.S. and Ph.D. are discussed in separate sections.

Please see the Graduate Coordinator for additional material regarding administrative guidelines, procedures, and help. It is the responsibility of the student to initiate processing the various forms; however, the School of Computing and the Graduate School will try to help you in any way possible.

2.1 The Graduate Studies Committee

The functions of the Graduate Studies Committee are as follows:

1. Oversee the graduate curriculum and make timely recommendations to the faculty on its findings.
2. Execute the School graduate policy on admissions and financial support.
3. Make teaching assistantship assignments.
4. Monitor the progress of all graduate students in the School. This monitoring relies on annual progress reports from students and their advisors (supervisory committee chairs). The advisor of a student whose progress is questionable will be invited to discuss the student’s situation with the Committee. Appropriate actions according to accepted School guidelines result from the student reviewing process.

During 2002-2003, the Committee consists of Professors Ross Whitaker (Director), Al Davis, and Kris Sikorski.

Information specific to a particular semester are on the Graduate Studies web page: <http://www.cs.utah.edu/graduate-studies>.

2.2 The Graduate Admissions Committee

The Graduate Admissions Committee recruits new graduate students and oversees the process by which they apply, are admitted, and are accepted into the graduate programs. During 2002-2003, the Graduate Admissions Committee consists of Professors Ellen Riloff (Chair), Bob Kessler, Kris Sikorski, Konrad Slind, and Bill Thompson.

2.3 The Graduate Coordinator

The Graduate Coordinator is the point of contact for students needing help in the program with permissions, forms, or questions. The Graduate Coordinator for 2002-2003 is Marilyn Gorder .

2.4 Financial Assistance For M.S. and Ph.D. Students

Assistantships and Fellowships

There are three types of financial aid available to graduate students in the School of Computing. Teaching and research positions are awarded on a semester-by-semester basis. Teaching assistantships are provided by the School, while research assistantships are awarded by individual faculty serving as investigators on research grants and contracts. A third form of support is research fellowships (including traineeships), the terms of which vary.

The duties and benefits of School teaching and research assistantships are defined as follows:

1. *Teaching Assistantship:* A teaching assistant is a matriculated student employed 10 to 20 hours per week to assist a faculty member in teaching. The teaching assistant is required to meet with students regularly in a classroom, laboratory, or other instructional setting; to assist in instructional duties through lesson and materials preparation; to counsel students outside of the regularly scheduled instructional periods; and to evaluate and grade students' work to aid in the determination of final course grades. Tuition waivers supplement salary support.
2. *Research Assistantship:* The Director of Graduate Studies assigns each research assistant to a particular faculty member based on mutual agreement of the faculty and student. The duties assigned to a research assistant are consonant with the student's research interests and also useful to the professor's research efforts. A research assistantship can be viewed as an internship, whereby the student learns by practicing under faculty supervision. Research assistants are employed up to 20 hours per week. A student wishing to be a research assistant should inquire directly with appropriate faculty sponsors. Tuition waivers supplement salary support.

Continuation of financial aid is dependent upon continued competent performance of teaching or research duties, as well as satisfactory progress in the student's program of study.

Research fellowships, such as those awarded by national foundations, can be requested directly from the granting agency under the supervision of the Director of Graduate Studies. In addition, the School annually nominates outstanding graduate students for University Research Fellowships, as well as fellowships for certain categories of graduate students from private corporations and federal agencies. In the past, students in the School have been awarded fellowships from NSF, DOE, DARPA, ACM, AMOCO, Apple, ARO, ONR, and IBM.

Summer Support

Faculty members who are conducting research projects often hire students as research assistants from a half to a full-time basis during the summer. Students interested in a summer appointment as a research assistant should make arrangements directly with the appropriate faculty member. There are a few teaching assistantships available during the Summer Semester.

Graduate Assistantship Duration and Pay Scales

Graduate student support is contingent upon satisfactory progress toward degree completion. The number of assistantships in each semester depends upon available funding. The School will notify students of appointments as early as possible before the start of a semester. It is usually not possible, however, to notify students of their specific assignments (that is, whether they will be an RA or TA) until the beginning of each semester, as these decisions are based on class enrollment figures. The table below gives the current pay scale for School of Computing graduate RAs and TAs at the customary 20 hour/week level. Undergraduate TAs are paid on a separate hourly scale.

Description	Level of Support
Student in M.S. program	\$1,400/mo
Student in Ph.D. program (pre-proposal)	\$1,400/mo
Student in Ph.D. program (post-proposal)	\$1,500/mo

An additional \$75/mo is provided during the academic year to cover the costs of student health insurance.

Full-time students who are employed by the University are exempt from FICA (Social Security) taxes, and should file a form to this effect. Students who are funded as RA's and TA's during the summer must register for the minimum of 3 required hours in order to be eligible for FICA exemptions. This form is available from the School administrator or from the Payroll office in 103 Park Building. A form must be filed *each* semester to keep the exemption in effect.

Tuition Waivers

Tuition waivers are available to Teaching and Research Assistants working at least 20 but not more than 29.6 hours/week. Tuition benefits are applicable to graduate-level courses, i.e., those numbered 5000 and above, that contribute to the student's degree course requirements. Any student wishing to take an undergraduate-level course will have to pay tuition for that course. Students must take between 9 and 12 semester hours to receive a full tuition waiver.

By Graduate School policy, tuition waivers are limited to four semesters for M.S. students. Students entering the Ph.D. program with a master's degree in Computer Science are limited to six semesters of tuition waiver support. Ph.D. students entering without a master's are limited to ten semesters of tuition waiver support. Ph.D. students with a master's degree in other areas must petition for ten semesters of tuition waiver support when entering the program. See the Graduate School Bulletin for additional requirements.

Procedures for Applying for Financial Assistance

Qualified full-time graduate students are eligible for financial aid in the form of teaching assistantships or research assistantships as described above. For incoming students, a financial assistance application form can be obtained in the School of Computing as well as from the Graduate Fellowship Office, 312 Park Building. A completed application should be submitted directly to the School of Computing by February 1 for aid to commence the following Fall Semester.

2.5 Transfer Credit

2.5.1 All Students

A student may not count more than eight semester hours of non-matriculated graduate work toward any graduate degree unless the student's registration for more than eight semester hours is specifically approved in advance by the School Director and the Dean of the Graduate School. Graduate courses taken as an undergraduate at the University of Utah cannot be counted towards a degree program unless a petition for graduate credit was filed with the University's Registrar at the time the course was taken.

Students who have done graduate study at other institutions may transfer up to 6 credits to the University of Utah. The following guidelines apply:

1. The courses must be *bona fide* graduate level class work (e.g., independent study is excluded), with grade B- or better.
2. Students must be able to show that the course work was used toward any other degree.
3. Quarter hours are converted to equivalent semester hours by a .66 multiplicative factor (e.g., 9 quarter hours = 6 semester hours).
4. Approval of each course is granted by the student's supervisory committee and graduate studies. Course appropriateness is determined by consideration of course content and the student's declared research area.
5. Approved courses are certified by a transfer credit form.
6. Approval of a course taken elsewhere for transfer credit does not imply fulfillment of any specific required course. For more information on this see Sections 2.9 and 2.10.

2.5.2 Ph.D. Students

Ph.D. students may count up to 20 hours of coursework from other graduate degrees toward the coursework requirements associated with the program of study. The following guidelines apply.

1. Approved courses are certified by inclusion on the student's *Program of Study for the Ph.D. Degree*. All coursework on the program of study is subject to approval by the student's supervisory committee and the Director of Graduate Studies. The Program of Study must be submitted by the end of the student's second semester of study.
2. Approval of courses taken elsewhere cannot be counted directly toward the core course requirements, and they do not affect the qualifying exam requirements. Some students may elect to complete a course at Utah even though they may have taken a comparable course elsewhere, to aid in preparation for these exams.
3. Approval of courses taken elsewhere has no effect on University of Utah residency requirements (see Section 2.10).

2.6 Leave of Absence Policy

If a student does not plan to take classes during a fall or spring semester, a leave of absence must be requested. Contact the Graduate Coordinator for the proper form.

2.7 GPA Requirements

Students are required to meet minimum grade and GPA requirements for their classes. Classes on the Program of Study must have a course number of 5000 or above. Students are required to earn a grade of B- or better in every class on the Program of Study, and must maintain an overall GPA of at least 3.0 in those classes. Students are required to earn a grade of B or better in any required class, and achieve an overall GPA of 3.5 in required classes.

2.8 Course Load

Full-time graduate students in the School of Computing are ordinarily requested to register for 12 hours, which includes regular courses, seminars, and research credits as appropriate. This is especially the case for students being supported via research or teaching assistantships.

Graduate School policy dictates that a graduate student who receives a full tuition waiver during any semester in which he or she holds an assistantship, fellowship or traineeship is required to register for at least nine semester hours, including thesis research and seminars.

All teaching assistants, as well as students receiving fellowship or traineeships stipends, are required to register for CS 6930 (Computer Science Seminar).

Students must be registered for at least three hours per semester, exclusive of summer semester, in order to remain in a graduate degree program. Students who do not maintain continuous registration and who have not been granted a leave of absence by the Graduate School are subject to being discharged from the degree program.

Students must be registered for at least three semester hours during the semester of the student's thesis defense. Once a student has passed the thesis defense, the student does not have to register the next term if within the 90-day period to turn in the final thesis.

2.9 The M.S. Degree in Computer Science

The M.S. in Computer Science is a research degree offered through the Graduate School. A student who has been accepted by the Graduate School is formally admitted to candidacy for the M.S. degree at the recommendation of the student's supervisory committee. Admission to candidacy occurs after the student:

- Forms a supervisory committee,
- Files an approved Program of Study form,
- Passes the comprehensive examination, and
- Submits an approved thesis proposal.

An application for candidacy must be submitted to the Graduate School no later than the last day preceding the semester of graduation. For the degree to be conferred, the approved Program of Study form must be completed and the thesis completed and publicly defended.

Each of these steps is described below. Most of the steps involve completing and submitting a properly signed form. Forms and assistance are available from the Graduate Coordinator.

Supervisory Committee. An M.S. committee consists of three members. A committee typically consists of School faculty, but may include one qualified external member. The committee should be formed by the second semester of enrollment in the M.S. program.

Any School of Computing regular faculty member may serve as a supervisory committee chair. Research or adjunct faculty may chair supervisory committees if accorded that privilege by the regular faculty. Individuals who are not faculty members may serve on supervisory committees if nominated by the regular faculty on the committee, and endorsed by the Graduate Studies Committee and School Director.

Final approval of all supervisory committees must be granted by the Dean of the Graduate School. M.S. students must form this committee by the end of the second semester of study, although a committee may be revised later by petition to the Graduate Studies Committee.

Required Courses. For the M.S. program the School of Computing expects incoming students to have demonstrated a basic understanding of fundamental concepts in Computer Science. The curriculum requirements for M.S. students are designed to ensure that all students who receive an M.S. degree have a working knowledge of those topics in computer science that are deemed fundamental by the faculty. This comprises a basic education in the core areas of computer science and a deeper education in one or more areas in which they will perform research. A core area illuminates the ways that most applications are designed and implemented, examines the hardware and software systems that are needed to execute the applications, and analyzes the resulting performance of such applications. M.S. students are required to complete courses in each of these areas.

To satisfy these goals M.S. students must take the following *required* courses:

- CS 6100 (Foundations of Computer Science)
- CS 5460 (Operating Systems)
- CS 6810 (Advanced Computer Architecture)

M.S. students are required to earn a grade of B or better in each of these 3 required classes, and achieve an overall GPA of B+ (numerical equivalent) in those classes.

Students may obtain a waiver for any of these required courses by demonstrating prior knowledge (e.g., completion of a similar course taken at another University). This waiver is obtained by petitioning the Graduate Studies Committee. The waiver procedure should be initiated by first contacting the Graduate Coordinator.

Program of Study. Course work listed on the approved Program of Study form must consist of at least 30 semester hours of graduate course work and thesis research. At least 6 semester hours of thesis research (CS 6970) and 20 semester hours of graduate course work must be included.

CS courses on the Program of Study must be numbered 6000 or above, excluding research credits (CS 6970); or they must be required courses. Of the required 20 semester hours, up to 6 may be in graduate courses outside of CS. At least one course in the Program of Study must be a CS course numbered 6000 or above, excluding independent study, seminars, research credit, and required courses.

At least 24 semester hours must be completed in resident study at the University of Utah. Students must be registered for a minimum of 3 semester hours during the semester in which the thesis is defended.

Independent study (CS 6950 and CS 7950) can be included in the Program of Study for the M.S. degree. Independent study for M.S. students will be allowed only when the project is self-contained and independent of thesis research. Course CS 6930 (Computer Science Seminar) may not be applied to the course requirements of the M.S. degree program.

A student may register for CS 6020 if that student writes and publishes a peer-reviewed article based on research performed in the University of Utah School of Computing. The contribution of the student to the article should be equivalent to that

conferred by first authorship. The paper should be published in a respectable outlet. It is the responsibility of the student's advisor to determine whether the student has made such a contribution, and whether the outlet is of sufficient quality. This paper must be accepted for publication prior to the end of the second year of study.

The Program of Study form should be filed with the School in the second semester of study and with the Graduate School prior to taking the comprehensive examination. The Program of Study form must be submitted to the Graduate School by the last day of the semester preceding the semester of graduation.

Thesis Proposal. The student should prepare and receive approval for a thesis proposal by the end of the third semester of study (not counting summers). A copy of the thesis proposal must be in the student's file. For guidelines on preparing proposals, consult *Discussion on Ph.D. Thesis Proposals in Computing Science*, by H. C. Lauer. Copies are available from the Graduate Coordinator or from the Thesis Editor. The thesis proposal must be approved at least one semester prior to the semester of the thesis defense.

Comprehensive Examination. The comprehensive examination for M.S. students is coupled to the thesis proposal. It consists of an oral examination on the thesis proposal and research area in a very broad sense. This examination is administered by the student's supervisory committee and should be completed by the end of the student's third semester of study (not counting summers) as a graduate student in the School. The examination should serve as the defense of the student's thesis proposal as well as to establish competence in the research area. The examination must be completed at least one semester prior to the semester of the thesis defense.

Completing Program of Study. An M.S. student is expected to devote the necessary time to courses and research in order to make satisfactory progress toward the degree. Satisfactory progress includes personal participation in the research and teaching environment of the School on a day-to-day basis.

A full time student working on an M.S. program is expected to complete the degree requirements within two calendar years. Beyond this period a student generally does not receive graduate financial support from the School, and the tuition waiver does not apply. A student must petition the Graduate Studies Committee to continue beyond the third year. The Graduate School limits M.S. programs to four years.

The requirements for the thesis defense and thesis are common with the Ph.D. program, and are discussed in a later section.

2.10 The Ph.D. Degree in Computer Science

The Ph.D. in Computer Science is a research degree offered through the Graduate School. It is awarded to a candidate who has demonstrated breadth in the areas represented by the School of Computing in general, and depth in a research specialty within the School of Computing. The latter is exhibited through the writing and defense of a dissertation that reports substantial original contributions in an approved area of research.

A student who has been accepted by the Graduate School is formally admitted to candidacy for the Ph.D. by the University at the recommendation of the student's supervisory committee. Admission to candidacy occurs after the student:

- Forms a supervisory committee,
- Files an approved Program of Study form,
- Completes the core course requirements,
- Passes the written portion of the qualifying examination,
- Passes the oral portion of the qualifying examination, and

- Submits an approved dissertation proposal.

An application for candidacy must be submitted to the Graduate School no later than the last day preceding the semester of graduation. For the degree to be conferred, the approved Program of Study form must be completed and the dissertation completed and publicly defended.

Each of these steps is described below. Most of the steps involve completing and submitting a properly signed form. Forms and assistance are available from the Graduate Coordinator.

Supervisory Committee. Each student forms a supervisory committee whose members guide the student's research program. The committee conducts the student's written qualifying examination, oral qualifying examination, and dissertation defense. A Ph.D. supervisory committee consists of five faculty members. At least three faculty members must be from the School, and at least one member from outside the School of Computing. Any School of Computing regular faculty member may serve as a supervisory committee chair. Research or adjunct faculty may chair supervisory committees if accorded that privilege by the regular faculty. Individuals who are not faculty members may serve on supervisory committees if nominated by the regular faculty on the committee, and endorsed by the Graduate Studies Committee and School Director.

Final approval of all supervisory committees is granted by the Dean of the Graduate School. Students must form this committee by the end of the second semester of study, although a committee may be revised later by petition to the Graduate Studies Committee.

Required Courses. Ph.D. students must demonstrate core knowledge in computer science by passing 5 specified courses, prior to the start of their fifth semester of study, with grades of B or better in each course and an overall GPA in the specified courses greater than 3.5. The specified courses consist of CS 6100 (Foundations of Computer Science), CS 5460 (Operating Systems), and CS 6810 (Advanced Computer Architecture), plus two of the following four courses: CS 5470 (Compiler Principles and Techniques), CS 6480 (Data Communications and Networking), CS 6520 (Programming Languages and Semantics), and CS 6210 (Advanced Scientific Computing). At most 9 credits of the 27 semester hours of regular graduate course work required of Ph.D. candidates can consist of the specified courses listed above.

Students may *not place out* of this requirement by substituting or transferring courses from other institutions. However, with approval of the Graduate Studies Committee, a student may replace one or more of these courses with a more advanced course in the same or related subject areas. Substitute courses must be "regular" classes with exams and/or assignments, not seminar, readings, or independent study classes. The Graduate Studies Committee has a list of pre-approved substitutions given in in Figure ???. Other substitutions require approval ahead of time. Notice that a student must have 5 distinct courses that satisfy the Ph.D. course requirements. Each advanced course can be offered as a substitute for only one required course.

Program of Study. Course work listed on the approved Program of Study form must comprise at least 50 semester hours of graduate course work and dissertation research, exclusive of independent study. Graduate course work applied toward an M.S. degree may be included. At least 14 semester hours of dissertation research (CS 7970) and 27 semester hours of graduate course work must be included. Up to 20 hours of graduate level coursework already applied to other degrees may be used in the program of study. See Section 2.5 for more information on this.

CS courses on the Program of Study must be at the 6000 level or above, excluding independent study, seminars, and research credits. Of the required 27 semester hours, up to 6 may be graduate courses outside of CS.

Neither courses CS 6930–CS 6944 (Computer Science Seminars) nor Independent study (CS 6950 and CS 7950) can be included in the Program of Study for the Ph.D. degree.

One year of study must be spent in full-time residency at the University (i.e., the student must enroll for a minimum of nine hours per semester for two consecutive semesters, summer optionally excluded). After the residency requirement is fulfilled, registration for three semester hours of CS 7970 (Ph.D. Dissertation Research) is considered a full load.

A student may register for CS 6020 if that student writes and publishes a peer-reviewed article based on research performed in the University of Utah School of Computing. The contribution of the student to the article should be equivalent to that conferred by first authorship. The paper should be published in a respectable outlet. It is the responsibility of the student's advisor to determine whether the student has made such a contribution, and whether the outlet is of sufficient quality. This paper must be accepted for publication prior to the end of the second year of study.

The Program of Study form should be filed with the School in the second semester of study and with the Graduate School prior to taking the qualifying examination. The Program of Study form must be submitted to the Graduate Records Office no later than the last day of the semester preceding the semester of graduation.

Dissertation Proposal. The student should prepare and receive approval for a dissertation proposal by the end of the sixth semester of study (not counting summers). A copy of the dissertation proposal must be in the student's file. For guidelines on preparing proposals, consult *Discussion on Ph.D. Thesis Proposals in Computing Science*, by H. C. Lauer. Copies are available from the Graduate Coordinator or from the Thesis Editor. The dissertation proposal must be approved at least one semester prior to the semester of the dissertation defense.

Qualifying Examination. After passing the Comprehensive Examination, all Ph.D. students must pass a Qualifying Examination, as specified by the Graduate School. The Qualifying Exam consists of a written part, to be conducted first, and an oral part.

The written part of the Qualifying Examination will cover the candidate's general area of specialization in sufficient depth to demonstrate his/her preparation for conducting Ph.D.-level research. Each member of the student's supervisory committee will contribute one or more questions to this exam. The supervisory committee will provide a written evaluation of this part of the exam, including an indication of whether or not the student will be allowed to proceed to the oral part of the Qualifying Examination.

The oral part comprises the dissertation proposal defense. At the supervisory committee's option, it may also include follow-up questions relating to the written part of the exam. All members of the supervisory committee should certify that the proposal is ready to be defended prior to conducting the oral part of the Qualifying Exam.

Students should pass their Qualifying Examination by the end of their sixth semester of study, not counting summer enrollment. The Qualifying Examination must be completed no less than one semester prior to defense of the dissertation.

Completing Program of Study. A Ph.D. student is expected to devote the necessary time to courses and research in order to make satisfactory progress toward the degree. Satisfactory progress includes personal participation in the research and teaching environment of the School on a day-to-day basis.

Dissertation. The completed dissertation must be published either in its entirety (through a legitimate publisher of the student's choice or through University Microfilms) or as one or more articles accepted for publication in approved scholarly journals. An abstract of each dissertation must be published in University Microfilms' Dissertation Abstracts International. Other requirements are common with the M.S. degree program and are discussed below.

2.11 M.S. Thesis and Ph.D. Dissertation Requirements

The supervisory committee must give preliminary approval of the thesis or dissertation prior to the defense. The defense can be scheduled after this approval. To schedule the defense, contact the Graduate Coordinator. Students are strongly encouraged to schedule the defense during a regular colloquium slot.

The student must provide one copy of the thesis or dissertation to the chair of the supervisory committee at least three weeks before the defense, and one copy to each of the other committee members at least two weeks prior to the defense. A complete draft of the thesis or dissertation must be delivered to the Graduate Coordinator one week prior to the announced

time of defense. This copy will be made available for public access. Students are encouraged to place an additional copy on the School of Computing web pages at least one week prior to the announced time of defense.

After successfully defending the thesis or dissertation, the student must obtain approval from the Final Reader (typically the supervisory committee chair), School Director, and Dean of the Graduate School. A draft of the final thesis or dissertation must then be presented to the Thesis Editor. Successful completion of the defense must be reported to the Graduate School at least four weeks before the last day of examinations in the final semester. Students should also read the document regarding copyright notices provided by the School and declare their intentions regarding granting the School the right to photocopy the thesis or dissertation before notifying the Graduate Coordinator of completion of the defense.

The student has one month after the defense to make any revisions prior to submitting the thesis or dissertation to the Graduate School Thesis Editor. There will be at most two additional months to complete any changes required by the Thesis Editor before final acceptance. If either of these deadlines are not met, the candidate must redo the oral defense. The final thesis or dissertation must be filed one week before the end of the semester of graduation.

Students are expected to offer each committee member a bound copy of the thesis or dissertation once it is completed. Detailed policies and procedures concerning the thesis or dissertation are contained in "A Handbook for Theses and Dissertations" published by the Graduate School.

2.12 Thesis or Dissertation Copyright Policy

The School of Computing has a fiduciary interest in seeing that the results of its research programs are widely disseminated so that other researchers and the public as a whole can benefit. This is especially true for work supported in part by governmental agencies. Copyrighting of a thesis or dissertation, without making any provision for its legal reprinting, severely limits the dissemination of research results. It also makes it difficult for the School to handle requests for copies of theses or dissertations received from other universities or researchers.

The School of Computing encourages distribution of theses and dissertations in any of the following ways:

- Through an appropriate agreement with University Microfilms, Inc., that allows them to reproduce the thesis on request (this option is only open to Ph.D. dissertations at the present time).
- Give permission to the School to have the thesis or dissertation reproduced upon request, with the School allowed to recover its costs of reproduction and distribution.
- Furnish the School with the address of the person, agency, or company that will handle the distribution of the thesis or dissertation.

2.13 The M. Phil. Degree

This degree requires the same qualifications for admission and scholarly achievement as the Ph.D, but does not require a dissertation and requires 54 semester hours of course work. All regulations covering the Ph.D. degree apply to the M.Phil. degree. This degree, like the Ph.D. degree, is a terminal degree; a student cannot be a candidate for both degrees in the this School. The School considers applications for admission to the M.Phil. program only from students already matriculated in the Ph.D. program.

3

Computing Facilities

The School of Computing provides state of the art computing facilities for both instructional and research use. Both facilities share a common network infrastructure that is based on an ATM fabric running at OC-12 (622 Mbps) and that provides desktop connections at speeds ranging from Fast Ethernet (100 Mbps) up to Gigabit Ethernet where necessary. The School's network attaches via an OC-12 connection directly to the campus OC-48 ATM mesh, which in turn routes traffic to Abilene (Internet 2), vBNS, and the Internet.

In addition to the shared network infrastructure, the core School of Computing facility supplies many centralized services, including shared disk space (4 Terrabytes), time, web/cgi, ftp, firewall, backups, printing resources, and email. Most services run on Solaris-based hosts, ranging from Ultra 10's to an Enterprise 5000. Several large-scale Solaris and Linux machines are also made available for general use.

The instructional computing facility includes over 180 Unix, Linux, and Windows-based machines. Most of these machines are organized into three laboratories and the remainder are situated in graduate student offices. The NT Lab in EMCB 210 includes approximately 90 Pentium II-based PCs. The electronic classroom in MEB 3225 contains 30 Pentium III-based PCs arranged into a classroom configuration. The CES/Grad Lab in MEB 3161 contains nine SGI workstations. Students in the School of Computing also have access to the College of Engineering Workstation Laboratory, which consists of five servers, more than 100 Sun workstations, and approximately 10 Linux workstations.

The research computing facility is a heterogeneous mix of over 300 machines, including SGI, HP, Sun and Intel-based hardware. The research computing facility includes major laboratories devoted to computer-aided design and graphics, computer systems, asynchronous digital systems and VLSI, robotics and vision, scientific computing and imaging, and information retrieval and natural language processing. These research laboratories contain a wide array of specialized equipment, including

- an SGI Origin 3800 (32 processors);
- an SGI Origin 2000 Reality Monster (96 processors, 8 IR heads);
- a 200-node network testbed and emulation facility;
- an SGI Power Onyx (14 processors, 2 RE2 heads);
- a multi-source nonlinear video editing environment;
- a real-time signal processing lab;

- an image analysis lab;
- equipment for various types of custom hardware design;
- a Sarcos Dextrous Arm, Utah/MIT Dextrous Hand, and PUMA 560 robots; and
- a Sarcos Treadport locomotion interface, several SensAble Phantom haptic interfaces, Fakespace Responsive Workbench, nVision Datavisor HiRes, and a variety of position trackers.

The College of Engineering operates a research-scale integrated circuit (IC) fabrication facility that is used extensively by the School of Computing. Equipment for testing and debugging both internally and externally fabricated circuits is housed in an integrated circuit testing facility that contains state-of-the-art HP, Tektronix and Micromanipulator automated IC testing equipment.

4

Computer Science Courses

The number and title of each course is followed by the number of semester hours it carries, the semester(s) during which it is taught (F=fall, S=spring, U=summer), its prerequisites, its corequisites, and any courses with which it is cross-listed.

Where a course has both a 5000- and 6000-level number, the 5000-level version is intended for undergraduate and the 6000-level version for honors and graduate students. The two versions of the class will meet together, but extra work will be expected of honors and graduate students.

Current class schedules and registration information¹ are available on line.

1000 Engineering Computing (3, FS) Coreq.: CP SC 1010, MATH 1210

Meets with CP SC 1001. Introduction to programming principles and engineering problem solving via computational means using MATLAB (during the first half of the semester) and C (during the second half of the semester). Decomposition of programs into data representations, functions, and control structures. Clean programming practices are emphasized. The MATLAB portion of the course focuses on the implementation of physically-based models, data visualization via plotting, and selected numerical techniques. The C portion of the course introduces basic syntax and special features of the language for engineering implementations.

1001 Engineering Computing using MATLAB (1.5, FS) Coreq.: CP SC 1010, MATH 1210

Meets with CP SC 1000. Introduction to programming principles and engineering problem solving via computational means using MATLAB. Decomposition of programs into data representations, functions, and control structures. Focus on the implementation of physically-based models, data visualization via plotting, and selected numerical techniques. Clean programming practices are emphasized. (This is a half-semester course that meets with CP SC 1000.)

1010 Introduction to Unix (0.5, FSU)

An introduction to the Unix workstations used in the College of Engineering CADE Lab. Topics include the X Windows system, Unix shell commands, file system issues, text editing with Emacs, accessing the World Wide Web with Netscape, and electronic mail. Self-paced course using online teaching aids.

¹<http://www.acs.utah.edu/prod/bin/student>

1020 Introduction to Programming in C++ (3, U)

An introduction to essential programming concepts using C++. Laboratory practice.

1021 Introduction to Programming in Java (3, FSU)

An introduction to essential programming concepts using Java. Laboratory practice emphasizes object-oriented techniques and web-based application design.

1040 Creating Interactive Web Content (3, FSU)

Introduction to the essentials of web page design and object-oriented programming through the use of HTML and JavaScript to create interactive web pages. It is appropriate for any student who is comfortable using a computer to write a paper and browse the Web. This is a 100% online course that can be completed on any computer equipped with a recent version of Netscape Communicator or Internet Explorer.

1050 Social Aspects of a Digital World (2, S)

Social and policy aspects of computing, beginning with a review of the history and technology of the Internet. Privacy, intellectual property, ethics, electronic commerce, and computer crime. Concurrent enrollment in a companion 1000-level discussion course (such as CP SC 1051) is required.

1051 Introductory Discussion of Social Aspects (1, S)

The combination of CP SC 1050/1051 is appropriate for any student who is already comfortable using a computer to write papers and explore the Web.

1950 Independent Study (1 to 4)**1960 Special Topics (1 to 4)****2000 Introduction to Programming in C (4, F) Coreq.: CP SC 1010**

Introduction to essential programming concepts using C. Decomposition of programs into functional units; control structures; fundamental data structures of C; recursion; dynamic memory management; low-level programming. Some exposure to C++. Laboratory practice. (Intended for non-CS/CE majors).

2010 Introduction to Computer Science I (4, FS) Coreq.: MATH 1210, CP SC 1010

The first course required for students intending to major in computer science and computer engineering. Introduction to the engineering and mathematical skills required to effectively program computers, and to the range of issues confronted by computer scientists. Roles of procedural and data abstraction in decomposing programs into manageable pieces. Introduction to object-oriented programming. Extensive programming exercises that involve the application of elementary software engineering techniques.

2020 Introduction to Computer Science II (4, FS) Prereq.: CP SC 2010

The second course required for students intending to major in computer science and computer engineering. Introduction to the problem of engineering computational efficiency into programs. Classical algorithms (including sorting, searching, and graph traversal) and data structures (including stacks, queues, linked lists, trees, hash tables, and graphs). Analysis of program space and time requirements. Extensive programming exercises that require the application of elementary techniques from software engineering.

2100 Discrete Structures (3, FS) Prereq.: CP SC 2010

Introduction to propositional logic, predicate logic, formal logical arguments, finite sets, functions, relations, inductive proofs, recurrence relations, graphs, and their applications to Computer Science.

2950 Independent Study (1–4)**2960 Special Topics (1–4)**

3050 Social Aspects of a Digital World (2, F) Prereq.: Programming proficiency

Social and policy aspects of computing, beginning with a review of the history and technology of the Internet. Privacy, intellectual property, ethics, electronic commerce, and computer crime. Concurrent enrollment in a companion 3000-level discussion course (such as CP SC 3051) is required. (Not offered 2002–03.)

3051 Intermediate Discussion of Social Aspects (1, F) Prereq.: Programming proficiency

The combination of CP SC 3050/3051 is appropriate for students who have an understanding of computing technology comparable to that of a newly-admitted computer science major. (Not offered 2002–03.)

3100 Models of Computation (3, S) Quantitatively Intensive B.S. Course. Prereq.: CP SC 2020, CP SC 2100

Models of sequential computation, including finite-state automata, push-down automata, and Turing machines.

3200 Scientific Computation (3, F) Prereq.: CP SC 2020, MATH 2250

Scientific computation relevant to computer science and engineering; floating-point arithmetic, systems of linear equations (direct and iterative techniques), nonlinear equations (univariate and multivariate), interpolation and differentiation (divided differences), integration (mechanical and Gaussian quadratures, optimal quadratures), approximation by spline functions (natural splines and B-splines, optimality of splines).

3500 Software Practice (4, FS) Prereq.: CP SC 2020

Meets with CP SC 5010. Practical exposure to the process of creating large software systems, including requirements specifications, design, implementation, testing, and maintenance. Emphasis on software process, software tools (debuggers, profilers, source code repositories, test harnesses), software engineering techniques (time management, code and documentation standards, source code management, object-oriented analysis and design), and team development practice. Much of the work will be in groups and will involve modifying preexisting software systems.

3505 Honors Software Practice (4, F) Prereq.: CP SC 2020

Practical exposure to the process of creating large software systems, including requirements specifications, design, implementation, testing, and maintenance. Emphasis on software process, software tools (debuggers, profilers, source code repositories, test harnesses), software engineering techniques (time management, code and documentation standards, source code management, object-oriented analysis and design).

3510 Advanced Algorithms and Data Structures (4, FS) Quantitatively Intensive B.S. Course. Prereq.: CP SC 2020, CP SC 2100

Meets with CP SC 5020. Study of algorithms, data structures, and complexity analysis beyond the introductory treatment from CP SC 2020. Balanced trees, heaps, hash tables, string matching, graph algorithms, external sorting and searching. Dynamic programming, exhaustive search. Space and time complexity, derivation and solution of recurrence relations, complexity hierarchies, reducibility, NP completeness. Laboratory practice. (Not offered Fall 2002.)

3520 Programming Language Concepts (3, F) Prereq.: CP SC 3500, CP SC 3510

Ideas behind the design and implementation of programming languages. Syntactic description; scope and lifetime of variables; runtime stack organization; parsing and abstract syntax; semantic issues; type systems; programming paradigms; interpreters and compilers.

3700 Fundamentals of Digital System Design (4, S) Quantitatively Intensive B.S. Course. Cross-listed as ECE 3700. Prereq.: CP SC 2010, PHYCS 2220

Techniques for minimizing logic functions and designing common combinational circuits such as decoders, selectors, and adders. Synchronous and asynchronous sequential circuits, state diagrams, Mealy and Moore circuits, state minimization and assignment. Use of software tools for design, minimization, simulation, and schematic capture. Implementation with MSI, LSI, and field programmable gate arrays. Laboratory included.

3710 Computer Design Laboratory (3, F) Cross-listed as ECE 3710. Prereq.: CP SC/ECE 3700, CP SC/ECE 3810

Student groups design, build, and test a programmable device such as a computer or calculator.

3720 Analog & Digital Interfacing with Microprocessors & Microcontrollers (4, S) Cross-listed as ECE 3720. Prereq.: CP SC/ECE 3700

Fundamentals of digital-to-analog (D-to-A) and analog-to-digital (A-to-D) circuits, relays, stepper motors, and digital switches. Interfacing digital and analog circuits to computers and micro-controllers. Laboratory included.

3810 Computer Architecture (4, FS) Quantitatively Intensive B.S. Course. Cross-listed as ECE 3810. Prereq.: CP SC 2020

An in-depth study of computer architecture and design, from digital logic to operating systems, including topics such as pipelining, memory systems, parallel and serial communication, and interrupts. Performance measures and compilation issues. Computer architectures including RISC, CISC, stack, and parallel.

3950 Independent Study (1–4)**3960 Special Topics** (1–4)**3991 Computer Engineering Junior Seminar** (0.5, F) Cross-listed as ECE 3991. Prereq.: CE major status

Presentation from faculty and industry representatives to discuss trends in computer engineering, professionalism, ethics, the impact of engineering in global and societal contexts, lifelong learning, and contemporary issues.

3992 CE Prethesis (0.5, S) Cross-listed as ECE 3992. Prereq.: CP SC/ECE 3991, CE major status

Students do necessary library research, develop writing and speaking skills, and prepare and present a senior thesis proposal.

4010 Teaching Introductory Computer Science (1, FS) Prereq.: Permission of instructor

Issues confronted by undergraduate teaching assistants in introductory computer science courses, including leading lab sections, conducting office hours, grading assignments, communicating with students. Each student must currently be an undergraduate teaching assistant in the School of Computing. May be taken for credit up to three times.

4400 Computer Systems (3, S) Prereq.: CP SC 3500, CP SC 3810

Introduction to computer systems from a programmer's point of view. Machine level representations of programs, optimizing program performance, memory hierarchy, linking, exceptional control flow, measuring program performance, virtual memory, concurrent programming with threads, network programming.

4500 Software Engineering Laboratory (3, S) Prereq.: CP SC 3500, CP SC 3510, senior standing in Computer Science

Development of significant software systems by small student groups, with emphasis on applying sound, disciplined software engineering practice.

4540 Web Software Architecture (3, S) Prereq.: CP SC 3510

Software architectures, programming models, and programming environments pertinent to developing web applications. Topics include client-server model, multi-tier software architecture, client-side scripting (JavaScript), server-side programming (Servlets and JavaServer Pages), component reuse (JavaBeans), database connectivity (JDBC), and web servers.

4550 Simulation (3, F) Prereq.: CP SC 3500, CP SC 3510

Basic simulation modeling, modeling complex systems, basic probability and statistics for simulation, building valid simulations, random numbers, and output data analysis. Both discrete event and continuous simulation may be covered.

4710 Computer Engineering Senior Project (3, F) Cross-listed as ECE 4710. Prereq.: CP SC/ECE 3710, CP SC/ECE 3720, senior standing in Computer Engineering

Students design a microcomputer system that includes RAM, EPROM, and I/O devices. Capstone project for computer engineering majors. Formal written reports, one or more oral presentations.

4950 Independent Study (1–4)

4960–4964 Special Topics (1–4)

4970 Bachelor's Thesis (3) Prereq.: Senior standing in computer science

Only students who have previously worked with a faculty member in a research group may register for Bachelor's Thesis credit, and then only with the permission of the faculty member. An undergraduate thesis is a publication-quality description of work done in previous semesters. At a minimum a thesis must be published as a technical report; ideally, it should be submitted to a conference or journal. A Bachelor's Thesis is intended as an alternative to the senior Software engineering Laboratory for students who are headed for graduate school.

4991 CE Senior Thesis I (2, F) Cross-listed as ECE 4992. Prereq.: CP SC/ECE 3992 and approved senior thesis proposal

Students work on original senior thesis project.

4992 CE Senior Thesis II (2, S) Cross-listed as ECE 4992. Prereq.: CP SC/ECE 4991

Students work on original senior thesis project, make an oral presentation at the annual student technical conference, and prepare and submit their senior thesis for approval.

4999 Honors Thesis/Project (3) Upper-division Communications/Writing

Restricted to students in the Honors Program working on their Honors degree.

5010 Software Practice (4, FS) Prereq.: CP SC 2020 and permission of instructor

Meets with CP SC 3500. This course is for graduate students from other than the School of Computing. Practical exposure to the process of creating large software systems, including requirements specifications, design, implementation, testing, and maintenance. Emphasis on software process, software tools (debuggers, profilers, source code repositories, test harnesses), software engineering techniques (time management, code and documentation standards, source code management, object-oriented analysis and design), and team development practice. Much of the work will be in groups and will involve modifying preexisting software systems.

5020 Advanced Algorithms and Data Structures (3, FS) Prereq.: CP SC 5010 and permission of instructor

Meets with CP SC 3510. This course is for graduate students from other than the School of Computing. Study of algorithms, data structures, and complexity analysis beyond the introductory treatment from CP SC 2020. Balanced trees, heaps, hash tables, string matching, graph algorithms, external sorting and searching. Dynamic programming, exhaustive search. Space and time complexity, derivation and solution of recurrence relations, complexity hierarchies, reducibility, NP completeness. (Not offered Fall 2002.)

5050 Social Aspects of a Digital World (2, F) Prereq.: Permission of instructor

Social and policy aspects of computing, beginning with a review of the history and technology of the Internet. Privacy, intellectual property, ethics, electronic commerce, and computer crime. Concurrent enrollment in a companion 5000-level discussion course (such as CP SC 5051) is required. (Companion discussions may also be offered by other departments.) (Not offered 2002–03.)

5051 Advanced Discussion of Social Aspects (1, F) Prereq.: Permission of instructor

The combination of CP SC 5050/5051 is appropriate for students who have a graduate-level background in issues related to the social aspects of computing (e.g., intellectual property or electronic commerce). (Not offered 2002–03.)

5060 Legal Protection of Digital Information (2, F)

Ways of protecting digital information—computer software and databases—using intellectual property law. Copyrights, patents, trade secrets, and contracts as ways of protecting digital information.

5100 Foundations of Computer Science (3, F) Prereq.: CP SC 3100, CP SC 3500, CP SC 3510

Meets with CP SC 6100. Finite Automata and related topics (BDDs, Presburger Arithmetic, and decidable fragments of first-order logic). Automata on Infinite Words, connections with Specification and Verification of Systems. Push Down Automata, Turing Machines, Proofs by Reduction, Diagonalization, Problems in Computability. First-order Logic and Decidability. NP Completeness, P-space Completeness.

5210 Advanced Scientific Computing I (3, F) Prereq.: CP SC 3200, CP SC 3500, CP SC 3510, MATH 3160

Meets with CP SC 6210. An introduction to existing classical and modern numerical methods and their algorithmic development and efficient implementation. Topics include: numerical linear algebra, interpolation, approximation methods and parallel computation methods for nonlinear equations, ordinary differential equations, and partial differential equations.

5300 Artificial Intelligence (3, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 6300. Introduction to field of artificial intelligence, including heuristic programming, problem-solving, search, theorem proving, question answering, machine learning, pattern recognition, game playing, robotics, computer vision.

5310 Robotics (3, F) Cross-listed as ME EN 5220. Prereq.: CP SC 1000, MATH 2250, PHYCS 2220

Meets with CP SC 6310. The mechanics of robots, comprising kinematics, dynamics, and trajectories. Planar, spherical, and spatial transformations and displacements. Representing orientation: Euler angles, angle-axis, and quaternions. Velocity and acceleration: the Jacobian and screw theory. Inverse kinematics: solvability and singularities. Trajectory planning: joint interpolation and Cartesian trajectories. Statics of serial chain mechanisms. Inertial parameters, Newton-Euler equations, D'Alembert's principle. Recursive forward and inverse dynamics.

5320 Computer Vision (3, S) Prereq.: CP SC 3500, CP SC 3510, MATH 2210, MATH 2270

Meets with CP SC 6320. Basic pattern-recognition and image-analysis techniques, low-level representation, intrinsic images, "shape from" methods, segmentation, texture and motion analysis, and representation of 2-D and 3-D shape. (Not offered 2002–03.)

5340 Natural Language Processing (3, F) Prereq.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

Meets with CP SC 6340. Computational models and methods for understanding written text. Introduction to syntactic analysis, semantic analysis, discourse analysis, knowledge structures, and memory organization. A variety of approaches are covered, including conceptual dependency theory, connectionist methods, and statistical techniques. Applications include story understanding, fact extraction, and information retrieval.

5350 Machine Learning (3, S) Prereq.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

Meets with CP SC 6350. Techniques for developing computer systems that can acquire new knowledge automatically or adapt their behavior over time. Topics include concept learning, decision trees, evaluation functions, clustering methods, explanation-based learning, language learning, cognitive learning architectures, connectionist methods, reinforcement learning, genetic algorithms, hybrid methods, and discovery.

5460 Operating Systems (3, F) Prereq.: CP SC 3510, CP SC/ECE 3810, CP SC 4400

Characteristics, objectives, and issues concerning computer operating systems. Hardware/software interactions, process management, memory management, protection, synchronization, resource allocation, file systems, security, and distributed systems. Extensive systems programming.

5470 Compiler Principles and Techniques (3, S) Prereq.: CP SC 3100, CP SC 3510, CP SC/ECE 3810, CP SC 4400

Lexical analysis, top-down and bottom-up parsing, symbol tables, internal forms and intermediate languages, run-time environments, code generation, code optimization, semantic specifications, error detection and recovery. Use of software tools for lexical analysis and parsing.

5480 Data Communications and Networks (3, F) Prereq.: CP SC 3510, CP SC/ECE 3810, CP SC 4400

Meets with CP SC 6480. A comprehensive study of the principles and practices of data communication and networks. Topics include: transmission media, data encoding, local and wide area networking architectures, internetwork and transport protocols (e.g., IPv4, IPv6, TCP, UDP, RPC, SMTP), networking infrastructure (e.g., routers, name servers, gateways), network management, distributed applications, network security, and electronic commerce. Principles are put into practice via a number of programming projects.

5520 Anatomy of a Modern Programming Language (3, S) Prereq.: CP SC 3520

Requirements, challenges, and techniques for designing a modern programming language, currently focusing on Java as a case study. Syntactic and lexical issues, semantic specification, modularity concepts, support for object-oriented programming, types and subtypes, type safety and security, portability, compilability, dynamic linking and loading, program evolvability, use of meta data (reflection), multi-threading, native code generation and linkage, generic types, persistence.

5530 Database Systems (3, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 6530. Representing information about real world enterprises using important data models including the entity-relationship, relational and object-oriented approaches. Database design criteria, including normalization and integrity constraints. Implementation techniques using commercial database management system software. Selected advanced Topics such as distributed, temporal, active, and multi-media databases.

5540 Human/Computer Interaction (3, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 6540. Fundamentals of input/output devices, user interfaces, and human factors in the context of designing interactive applications.

5600 Introduction to Computer Graphics (3, S) Prereq.: CP SC 3500, MATH 2250; Coreq.: CP SC 3510 recommended

Basic display techniques, display devices, and graphics systems. Homogeneous coordinates, transformations, and clipping. Introduction to lighting models. Introduction to raster graphics and hidden-surface removal.

5605 Honors Introduction to Computer Graphics (3, S) Prereq.: CP SC 3505

Basic display techniques, display devices, and graphics systems. Homogeneous coordinates, transformations, and clipping. Introduction to lighting models. Introduction to raster graphics and hidden-surface removal.

5610 Advanced Computer Graphics I (3, F) Prereq.: CP SC 5600 or CP SC 3610

Meets with CP SC 6610. Interactive 3D computer graphics, polygonal representations of 3-D objects. Interactive lighting models. Introduction to interactive texture mapping, shadow generation, image-based techniques such as stencils, hidden-line removal, and silhouette edges. Introduction to image-based rendering, global illumination, and volume rendering.

5630 Scientific Visualization (3, F) Prereq.: CP SC 3500, CP SC 3510; CP SC 3200 or CP SC 5210 or MATH 5600

Meets with CP SC 6630. Introduction to the techniques and tools needed for the visual display of data. Students will explore many aspects of visualization, using a "from concepts to results" format. The course begins with an overview of the important issues involved in visualization, continues through an overview of graphics tools relating to visualization, and ends with instruction in the utilization and customization of a variety of scientific visualization software packages.

5710 Advanced Integrated Circuit Design I (3, F) Cross-listed as ECE 5710. Prereq.: CP SC/ECE 3700

Meets with CP SC 6710. Introduction to basic concepts of the design of CMOS integrated circuits for students with a wide range of backgrounds. Static and dynamic properties of CMOS circuits, composite layout of CMOS circuits, and modeling of transistors for use in SPICE simulations. Commonly encountered CMOS circuits. Introduction to CMOS analog/digital circuits. Students complete design, composite layout, and digitization of a simple integrated circuit using computer-aided design tools.

5720 Advanced Integrated Circuit Design II (3) Cross-listed as ECE 5720. Prereq.: CP SC/ECE 5710/6710, ECE 2100

Meets with CP SC 6720. Design of mixed signal (analog/digital) CMOS integrated circuits. Fundamental building blocks for analog circuits, including the basic principles of opamp, current mirror and comparator design. Basics of discrete-time signals and filters. Implementation of switched capacitor circuits and discussions of various implementations of D/A and A/D converters, oversampled converters and phase locked loops. (Not offered 2002–03.)

5740 Computer-Aided Design of Digital Circuits (3) Cross-listed as ECE 5740. Prereq.: CP SC/ECE 3700, CP SC 3510

Meets with CP SC 6740. Introduction to theory and algorithms used for computer-aided synthesis of digital integrated circuits. Topics include algorithms and representations for Boolean optimization, hardware modeling, combination logic optimization, sequential logic optimization and technology mapping. (Not offered 2002–03.)

5750 Synthesis and Verification of Asynchronous VLSI Systems (3, F) Cross-listed as ECE 5750. Prereq.: CP SC/ECE 3700, CP SC 3510

Meets with CP SC 6750. Introduction to systematic methods for the design of asynchronous VLSI systems from high-level specifications to efficient, reliable circuit implementations. Topics include specification, controller synthesis, optimization using timing information, technology mapping, data path design, and verification. (Not offered 2002–03.)

5810 Advanced Computer Architecture (3, F) Cross-listed as ECE 5810. Prereq.: CP SC/ECE 3700, CP SC/ECE 3810

Meets with CP SC 6810. Principles of modern high performance computer and micro architecture: static vs. dynamic issues, pipelining, control and data hazards, branch prediction and correlation, cache structure and policies, cost-performance and physical complexity analyses.

5830 VLSI Architecture (3) Cross-listed as ECE 5830. Prereq.: CP SC/ECE 3700, CP SC/ECE 3810

Meets with CP SC 6830. Project-based study of a variety of Topics related to VLSI systems. Use of field programmable gate arrays to design, implement, and test a VLSI project. (Not offered 2002-03.)

5940 Seminar (1-3)

Current Topics in computer science. May be repeated for credit.

5950 Independent Study (1–4)**5960–5969 Special Topics** (1–4)

The following special topics courses are currently scheduled for the 2002–03 academic year. Contact the faculty member in charge for details.

- **CP SC 5962 VLSI Logic Test, Validation, and Verification** (3,F). Prof. Kalla.
- **CP SC 5963 Advanced Manufacturing** (3, F). Prof. Drake.

6010 Writing Research Proposals (2, S) Prereq.: Graduate standing in Computer Science

Fundamental aspects of writing computer science research proposals, including thesis, dissertation, and grant proposals. Form, style, substance, and marketing of effective proposals will be considered. Emphasis is placed on developing and presenting clear and compelling ideas. Substantial writing and class presentations is required of all participants. (This is a half-semester course.)

6020 Conducting, Publishing, and Presenting Early-Career Research (3) Prereq.: Graduate standing in Computer Science

This is an independent study offering designed to encourage beginning graduate students to conduct, publish, and present original research early in their graduate careers. A graduate student can earn credit for CP SC 6020 by having a first-authored paper accepted for publication in a top-tier journal or conference and by subsequently presenting the published work in a one-hour research colloquium. The research must be conducted while a graduate student at Utah; the paper must be accepted within two years of enrolling in the graduate program; the journal or conference must be approved by the student's graduate committee; the colloquium must be presented as soon as possible after the acceptance of the paper; and the student must complete these requirements and register for CP SC 6020 within three years of enrolling in the graduate program. CP SC 6020 may not be repeated for credit.

6100 Foundations of Computer Science (3, F) Prereq.: CP SC 3100, CP SC 3500, CP SC 3510

Meets with CP SC 5100. Graduate and honors students only. Extra work required.

6110 Formal Methods for System Design (3, S) Prereq.: CP SC 5100/6100 and CP SC 6520

Study of methods for formally specifying and verifying computing systems. Specific techniques include explicit state enumeration, implicit state enumeration, automated decision procedures for first-order logic, and automated theorem proving. Examples selected from the areas of superscalar CPU design, parallel processor memory models, and synchronization and coordination protocols. (Not offered 2002–03.)

6210 Advanced Scientific Computing I (3, F) Prereq.: CP SC 3200, CP SC 3500, CP SC 3510, MATH 3160

Meets with CP SC 5210. Graduate and honors students only. Extra work required.

6220 Advanced Scientific Computing II (3, S) Prereq.: CP SC 5210/6210 or MATH 5600

A study of the numerical solution of two and three dimensional partial differential equations that arise in science and engineering problems. Topics include: finite difference methods, finite element methods, boundary element methods, multigrid methods, mesh generation, storage optimization methods, and adaptive methods.

6300 Artificial Intelligence (3, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 5300. Graduate and honors students only. Extra work required.

6310 Robotics (3, F) Cross-listed as ME EN 6220. Prereq.: CP SC 1000, MATH 2250, PHYCS 2220

Meets with CP SC 5310. Graduate and honors students only. Extra work required.

6320 Computer Vision (3, S) Prereq.: CP SC 3500, CP SC 3510, MATH 2210, MATH 2270

Meets with CP SC 5320. Graduate and honors students only. Extra work required. (Not offered 2002–03.)

6340 Natural Language Processing (3, F) Prereq.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

Meets with CP SC 5340. Graduate and honors students only. Extra work required.

6350 Machine Learning (3, S) Prereq.: CP SC 3500, CP SC 3510; CP SC 5300/6300 recommended

Meets with CP SC 5350. Graduate and honors students only. Extra work required.

6360 Virtual Reality (3, S) Prereq.: CP SC 5310/6310

Human interfaces: visual, auditory, haptic, and locomotory displays; position tracking and mapping. Computer hardware and software for the generation of virtual environments. Networking and communications. Telerobotics: remote manipulators and vehicles, low-level control, supervisory control, and real-time architectures. Applications: manufacturing, medicine, hazardous environments, and training. (Not offered 2002–03.)

6470 Advanced Topics in Compilation (3, F) Prereq.: CP SC 5470

Compilation of modern languages. Optimization techniques, register allocation and instruction scheduling, garbage collection, exception handling. Linkers and late-stage compilation and optimization. (Not offered 2002–03.)

6480 Data Communications and Networks (3, F) Prereq.: CP SC 3500, CP SC 3510, CP SC/ECE 3810

Meets with CP SC 5480. Graduate and honors students only. Extra work required.

6520 Programming Languages and Semantics (3, S) Prereq.: CP SC 3520, CP SC 3100

Examination of the formal and pragmatic ideas behind programming language design. Imperative, functional, logic, object-oriented, and multi-paradigm languages. Lambda calculus, fixpoints, type systems, and predicate logic. Denotational semantics and models of concurrency.

6530 Database Systems (3, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 5530. Graduate and honors students only. Extra work required.

6540 Human/Computer Interaction (3, F) Prereq.: CP SC 3500, CP SC 3510

Meets with CP SC 5540. Graduate and honors students only. Extra work required.

6610 Advanced Computer Graphics I (3, S) Prereq.: CP SC 5600 or CP SC 3610

Meets with CP SC 5610. Graduate and honors students only. Extra work required.

6620 Advanced Computer Graphics II (3, S) Prereq.: CP SC 5610/6610

Introduction to ray-tracing. Intersection methods for 3-D objects, reflection and refraction. Introduction to surface and solid texturing. Introduction to continuous-tone pictures and the aliasing problem. Special effects such as soft shadows, depth-of-field, motion-blur, and indirect lighting.

6630 Scientific Visualization (3, F) Prereq.: CP SC 3500, CP SC 3510; CP SC 3200 or CP SC 5210/6210 or MATH 5600

Meets with CP SC 5630. Graduate and honors students only. Extra work required.

6650 Image Synthesis (3, F) Prereq.: CP SC 5620/6620, CP SC 6670, MATH 5010

Using camera and sensor simulation along with physical simulation to generate realistic synthetic images. (Not offered 2002–03.)

6670 Computer-Aided Geometric Design I (3, F) Prereq.: MATH 2210, MATH 2250, CP SC 3500, CP SC 3510; Coreq.: CP SC 5600/6600**6680 Computer-Aided Geometric Design II** (3) Prereq.: CP SC 6670

Introduction to current concepts and issues in CAGD systems with emphasis on free-form surface design; mathematics of free-form curve and surface representations, including Coons patches, Bezier method, B-splines, triangular interpolants, and their geometric consequences; classical surface geometry; local and global design tradeoffs and explicit and parametric tradeoffs; subdivision and refinement as techniques in modeling; current production capabilities compared to advanced research. Laboratory experiments with current CAD systems. (Not offered 2002–03.)

6710 Advanced Integrated Circuit Design I (3, F) Cross-listed as ECE 6710. Prereq.: CP SC/ECE 3700

Meets with CP SC 5710. Graduate and honors students only. Extra work required.

6720 Advanced Integrated Circuit Design II (3) Cross-listed as ECE 6720. Prereq.: CP SC/ECE 5710/6710, ECE 2100

Meets with CP SC 5720. Graduate and honors students only. Extra work required. (Not offered 2002–03.)

- 6740 Computer-Aided Design of Digital Circuits** (3) Cross-listed as ECE 6740. Prereq.: CP SC/ECE 3700, CP SC 3510
Meets with CP SC 5740. Graduate and honors students only. Extra work required. (Not offered 2002–03.)
- 6750 Synthesis and Verification of Asynchronous VLSI Systems** (3, F) Cross-listed as ECE 6750. Prereq.: CP SC/ECE 3700, CP SC 3510
Meets with CP SC 5750. Graduate and honors students only. Extra work required. (Not offered 2002–03.)
- 6770 Advanced Digital VLSI Systems Design** (3) Cross-listed as ECE 6770. Prereq.: CP SC/ECE 5710/6710
Full custom, high speed, high performance CMOS circuit design issues, methodologies, and techniques. Failure modes, modeling techniques, testing, clock skew analysis, clock distribution, power analysis, power line distribution, electrical rules checking, megacell design flow, and other important design issues. (Not offered 2002–03.)
- 6810 Advanced Computer Architecture** (3, F) Cross-listed as ECE 6810. Prereq.: CP SC/ECE 3700, CP SC/ECE 3810
Meets with CP SC 5810. Graduate and honors students only. Extra work required.
- 6820 Parallel Computer Architecture** (3) Cross-listed as ECE 6820. Prereq.: CP SC/ECE 5810/6810
Architecture, design, and analysis of parallel computer systems: vector processing, data vs. control concurrency, shared memory, message passing, communication fabrics, case studies of current high performance parallel systems. (Not offered 2002–03.)
- 6830 VLSI Architecture** (3) Cross-listed as ECE 6830. Prereq.: CP SC/ECE 3700, CP SC/ECE 3810
Meets with CP SC 5830. Graduate and honors students only. Extra work required. (Not offered 2002–03.)
- 6930–6944 Seminar** (1-3)
Current Topics in Computer Science. May be repeated for credit.
- 6950 Independent Study** (1–4)
- 6960–6969 Special Topics** (1–4)
The following special topics courses are currently scheduled for the 2002–2003 academic year. Contact the instructor for details.
- **CP SC 6960 Advanced Networking** (3,F). Prof. Hsieh.
 - **CP SC 6962 VLSI Logic Test, Validation, and Verification** (3,F). Prof. Kalla.
 - **CP SC 6964 Image Processing for Graphic and Vision** (3,S). Prof. Whitaker.
- 6970 Masters Thesis Research** (1–12)
- 6980 Faculty Consultation Masters** (1–12)
- 7120 Information-Based Complexity** (3) Prereq.: CP SC 3200, MATH 2270, MATH 3210
Analysis of optimal computational methods for continuous problems. Introduction to the general worst case theory of optimal algorithms, linear problems, and spline algorithms as well as selected nonlinear problems. Examples include optimal integration, approximation, nonlinear zero finding, and fixed points. (Not offered 2002–03.)
- 7240 Sinc Methods** (3, S) Prereq.: CP SC 5210/6210 or MATH 5600 or MATH 5610
Sinc methods for solving difficult computational problems, such as partial differential and integral equation problems, that arise in science and engineering research. Emphasis on parallel computation. Applications vary, depending on participants in the class. Students are given projects—whenever possible in their areas of research—that lead to publishable research articles. (Not offered 2002–03.)

7310 Advanced Robotics (3, S) Cross-listed as ME EN 7230. Prereq.: CP SC/ME EN 5310/6310 5220/6220

Covers the kinematics, dynamics, and control of robotic manipulators. Projects controlling robots will be an integral part of the course.

7460 Advanced Operating Systems (3) Prereq.: CP SC 5460, CP SC 5480/6480

Practical distributed operating systems concepts from basics through the state of the art. Topics include interprocess communication, client-server systems, distributed shared memory, distributed file systems, distributed databases, portable computing, software fault tolerance, and wide-area (e.g. web) applications. Work includes individual oral presentations, a group project, and a written research report. (Not offered 2002–03.)

7940 Seminar (1–3)

May be repeated for credit.

7950 Independent Study (1–4)**7960 Special Topics** (1–4)

The following special topics courses are currently scheduled for the 2002–2003 academic year. Contact the instructor for details.

- **CP SC 7961 Vision Science** (3,S). Prof. Thompson.

7970 PhD Dissertation Research (1–12)**7980 Faculty Consultation PhD** (1–12)**7990 Continuing Registration: PhD** (0)

5

School of Computing Faculty and Their Research Interests

Erik Brunvand

Associate Professor, School of Computing
Ph.D., Carnegie Mellon University, 1991

Professor Brunvand¹ joined the faculty in 1990. He has interests in computer architecture and VLSI systems in general, and self-timed and asynchronous systems in particular. One aspect of his research involves compiling concurrent communicating programs into asynchronous VLSI circuits. The current system allows programs written in a subset of Occam, a concurrent message-passing programming language based on CSP, to be automatically compiled into a set of self-timed circuit modules suitable for manufacture as an integrated circuit. He is also interested in investigating the effects of asynchrony on computer systems architecture at a higher level. To explore these ideas he is building a series of prototype asynchronous computer systems out of FPGA and custom VLSI chips.

- Ganesh Gopalakrishnan, Prabhakar Kudva, and Erik Brunvand, "Peephole Optimization of Asynchronous Macro-module Networks," IEEE Transactions on VLSI Systems, Vol. 7, No 1, March 1999.
- William Richardson and Erik Brunvand, "Fred: A Decoupled Self-Timed Computer Architecture with Precise Exceptions," IEE Proceedings on Computers and Digital Techniques, Special issue on Asynchronous Processors. Vol. 143, No. 5, September 1996.
- Erik Brunvand, Steven Nowick, and Kenneth Yun, "Modern Asynchronous Circuit Design," TAU-99, Monterey, CA, March 1999.
- J. Carter, W. Hsieh, L. Stoller, M. Swanson, L. Zhang, E. Brunvand, A. Davis, C.-C. Kuo, R. Kuramkote, M. Parker, L. Schaelicke, and T. Tateyama, "Impulse: Building a Smarter Memory Controller," The Proceedings of the Fifth International Symposium on High Performance Computer Architecture, Jan 1999
- Ajay Khoche and Erik Brunvand, "Critical Hazard-Free Test Generation for Asynchronous Circuits," VLSI Test Symposium (VTS'97).
- William Richardson and Erik Brunvand, "Precise Exception Handling for a Self-Timed Processor," in IEEE International Conference on Computer Design (ICCD95) *Winner of best paper award in Design & Test track at ICCD95.*

¹<http://www.cs.utah.edu/~elb/>

John Carter

Associate Professor, School of Computing
Ph.D., Rice University, 1992

Professor Carter² joined the faculty in January 1993. His research interests include computer architecture, operating systems, distributed systems, and computer networks. Of particular interest are novel memory system designs, both hardware and software. Dr. Carter is co-leading two research projects: the Impulse Adaptable Memory Systems project and the Khazana project. The goal of the Impulse project is to attack the primary problem limiting performance in future computer systems – the inability of conventional memory systems to supply data fast enough to avoid processing stalls – by developing a main memory controller and associated software that allows applications to dynamically change the way that the processor’s memory hierarchy is managed. Khazana makes it easier for programmers to develop sophisticated distributed applications by addressing the shared state management problem faces by most such applications. Khazana exports the abstraction of a distributed secure persistent globally shared store that applications can use to store their shared state. It is responsible for performing many of the common operations needed by distributed applications, including replication, consistency management, and fault recovery.

- J.B. Carter, W.C. Hsieh, et al. Impulse: Building a Smarter Memory Controller. In the *Proceedings of the Fifth International Symposium on High Performance Computer Architecture*, pp. 70-79, January 1999.
- C.-C. Kuo, J.B. Carter, R. Kuramkote, and M. Swanson. AS-COMA: An Adaptive Hybrid Shared Memory Architecture. In the *Proceedings of the 1998 International Conference on Parallel Processing (ICPP’98)*, August 1998.
- M. Swanson, L.B. Stoller, and J.B. Carter. Increasing TLB Reach Using Superpages Backed by Shadow Memory. In the *Proceedings of the 25th Annual International Symposium on Computer Architecture*, June 1998, pp. 204-213.
- J.B. Carter, A. Ranganathan, and S. Susarla. Khazana: An Infrastructure for Building Distributed Services. In the *Proceedings of the 18th Annual International Conference on Distributed Computing Systems*, pp. 562-571, May 1998.
- J.B. Carter, J.K. Bennett, and W. Zwaenepoel. Techniques for reducing consistency-related communication in distributed shared memory systems. *ACM Transactions on Computer Systems*, August 1995.
- J.B. Carter, J.K. Bennett, and W. Zwaenepoel. Implementation and performance of Munin. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pages 152–164, October 1991.

Elaine Cohen

Professor, School of Computing
Adjunct Associate Professor of Mathematics
Ph.D., Syracuse University, 1974

Professor Cohen³ has held a faculty position since 1974. Currently she is co-head of the School’s Computer-Aided Geometric Design Group. Present research is centered around representational and algorithmic problems associated with geometric modeling, graphics, scientific visualization physically based modeling, process planning, CAD/CAM and CAE. Dr. Cohen received a B.A. in Mathematics from Vassar College in 1968 and M.S. and Ph.D. degrees in mathematics from Syracuse University in 1970 and 1974, respectively.

- T. V. Thompson II, D. E. Johnson, and E. Cohen,, ”Direct Haptic Rendering Of Sculptured Models,” in Proc. Symposium on Interactive 3D Graphics, (Providence, RI), pp. 167-176, ACM, April 1997.
- G. Elber and E. Cohen, ”Adaptive Iso-Curves Based Rendering for Free Form Surfaces,” *Transactions on Graphics*, 1996, v. 15, n. 3, pp.249 - 263.
- E. Driskill and E. Cohen, ”Interactive Design, Analysis, and Illustration of Assemblies,” in Proc. Symposium on Interactive 3D Graphics, pp. 27-32, ACM, April 1995.

²<http://www.cs.utah.edu/~retrac/>

³<http://www.cs.utah.edu/~cohen/>

- G. Elber and E. Cohen, “Arbitrarily Precise Computation of Gauss Maps and Visibility Sets,” in Proc. Solid Modeling 95, (Salt Lake City, UT), Solid Modeling, May 1995.
- R. Riesenfeld, E. Cohen, S. Drake, and L. Gursoz, “Modeling Issues in Solid Freeform Fabrication”, in Proc. NSF Solid Freeform Fabrication Workshop, 1995.

Al Davis

Professor, School of Computing
Ph.D., University of Utah, 1972

Professor Davis⁴ joined the faculty in 1993. His research interests involve high performance computer architectures and digital system design methodologies. More specifically he is interested in parallel processor architectures, high performance uniprocessor I/O architecture, VLSI, VLSI CAD, high performance communication, and asynchronous circuits. Prior to his joining the faculty in the fall of 1993, he spent the previous 12 years as a research scientist working on the design and implementation of parallel processing systems at Schlumberger Palo Alto Research and subsequently at Hewlett-Packard Laboratories. Recent accomplishments include 1) the development of an automatic asynchronous circuit synthesis system called STETSON; 2) the design and implementation of an asynchronous scalable parallel communication fabric VLSI component called FEDEX which is capable of supporting 500 MB/sec sustained bandwidth on each of its 7 ports; and 3) the development of an extensible and scalable parallel processing system called MAYFLY and 4) the development of a very low latency message passing protocols called Direct Deposit. He is currently involved in the DARPA sponsored Impulse project (Prof. John Carter is the PI) which is developing an adaptive memory controller that is capable of dynamically organizing cache lines to suit the applications needs. During the 1999-2000 academic year, Professor Davis was on sabbatical at Intel in Austin, Texas where he lead the I/O and memory architecture efforts for the IA32 processor which is expected to be in production in 2003.

- J. Carter, W. Hsieh, L. Stoller, M. Swanson, L. Zhang, E. Brunvand, A. Davis, C.-C. Kuo, R. Kuramkote, M. Parker, L. Schaelicke, and T. Tateyama. Impulse: Building a Smarter Memory Controller. Proceedings of the Fifth International Symposium on High Performance Computer Architecture, pp. 70-79, January 1999.
- L. Schaelicke and A. Davis. Improving I/O Performance with a Conditional Store Buffer. Proceedings of the 31st Annual ACM/IEEE International Symposium on Microarchitecture, pp. 160-169, November 1998.
- A. Davis, M. Swanson, M. Parker. Efficient Communication Mechanisms for Cluster Based Parallel Computing. Springer-Verlag Lecture Notes in Computer Science #1199. Feb. 1997, pp. 1-15.
- A. Davis. *Asynchronous Digital Circuit Design*. Chapter 3: ‘Synthesizing Asynchronous Circuits: Practice and Experience’. Springer-Verlag Workshops in Computing series, April 1995, pp. 104 – 151.
- A. Davis. R2 - A Damped Adaptive Router Design. Parallel Computer Routing and Communication. Springer-Verlag Lecture Notes in Computer Science #853. May 1994, pp. 295-309.
- A. Davis. Mayfly: A General-Purpose, Scalable, Parallel Processing Architecture. Lisp and Symbolic Computation, Volume 5, Numbers 1/2, May 1992, pp. 7-47.

Matthew Flatt

Assistant Professor, School of Computing
Ph.D., Rice University, 1999

Professor Flatt’s⁵ research interests cover all practical and theoretical aspects of programming languages, systems, and environments. As part of the Programming Languages Team (PLT), he is one of the principal architects of the DrScheme programming environment and a co-author of the *How to Design Programs* textbook. His current research topics include module languages for software components, object-oriented languages for classes and mixins, and high-level operating systems for cooperating applications.

⁴<http://www.cs.utah.edu/~ald/>

⁵<http://www.cs.utah.edu/~mflatt/>

- S. McDirmid, M. Flatt, and W. C. Hsieh. “Jiazzi: New-Age Components for Old-Fashioned Java.” In *Proc. ACM Conference on Object-Oriented Programming, Languages, Systems, and Applications*, October 2001.
- A. Reid, M. Flatt, L. Stoller, J. Lepreau, and E. Eide. “Knit: Component Composition for Systems Software.” In *Proc. USENIX Symposium on Operating Systems Design and Implementation*, October 2000.
- M. Flatt, R. B. Findler, S. Krishnamurthi, and M. Felleisen. “Programming Languages as Operating Systems (or, Revenge of the Son of the Lisp Machine).” In *Proc. ACM International Conference on Functional Programming*, September 1999.
- M. Flatt and M. Felleisen. “Cool Modules for HOT Languages.” In *Proc. ACM Conference on Programming Language Design and Implementation*, June 1998.
- R. B. Findler, C. Flanagan, M. Flatt, S. Krishnamurthi and M. Felleisen. “DrScheme: A Pedagogic Programming Environment for Scheme.” In *Proc. International Symposium on Programming Languages: Implementations, Logics, and Programs*, September 1997.

Ganesh C. Gopalakrishnan

Professor, School of Computing
Ph.D., State University of New York at Stony Brook, 1986

Professor Gopalakrishnan's⁶ primary research is in verification methods for concurrent systems such as shared memory systems, microprocessor busses, multithreaded software, and message passing networks. He also maintains active interest in self-timed design. Today's concurrent systems employ complex protocols that are expected to guarantee properties such as in-order arrival of messages, deadlock freedom, and liveness. In modern design approaches, these systems are subject to a battery of conventional tests, and when all these pass, model-checking methods are brought to bear to tracking down elusive bugs that may cripple the system well after field deployment. The effectiveness of conventional model-checking methods is limited by their inability to handle large state spaces, deal with parameterized designs, or provide guidelines for writing a comprehensive list of properties to check. Our group's recent efforts have addressed these problems using realistic driving problems such as generalized multi-level PCI I/O busses, the Intel Itanium Shared Memory Model, and the Java Shared Memory Model for Multithreading. Professor Gopalakrishnan was a general co-Chair of the Internal Symposium on Formal Methods in Computer-Aided Design (FMCAD) in November 1998, the International Symposium on Advanced Research in Asynchronous Circuits and Systems (Async) in November 1994. He organized the Workshop on Advances in Verification (WAVE) as well as the workshop on Formal Specification and Verification Methods for Shared Memory Systems, both in year 2000.

- Prosenjit Chatterjee and Ganesh Gopalakrishnan, "Towards a Formal Model of Shared Memory Consistency for Intel ItaniumTM," International Conference on Computer Design, October 2001.
- Yue Yang, Ganesh Gopalakrishnan, and Gary Lindstrom, "Analyzing the CRF Java Memory Model using Mur ϕ ," Workshop on Software Model Checking, July 2001 (Post CAV'01).
- Michael Jones and Ganesh Gopalakrishnan, "Verifying Transaction Ordering Properties in Unbounded Multi-Bus Networks through Algorithmic and Deductive Methods," Formal Methods in Computer-Aided Design, November 2000.
- Ravi Hosabettu, Ganesh Gopalakrishnan, and Mandayam Srivas, "Verifying Microarchitectures that Support Speculation and Exceptions," Computer Aided Verification, July 2000.
- Hans Jacobson, Erik Brunvand, Ganesh Gopalakrishnan, and Prabhakar Kudva, "High Level Asynchronous System Design Using the ACK Framework," Proc. Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, April 2000.

David H. Hanscom

Clinical Professor, School of Computing
Ph.D., Case Western Reserve University, 1970

Professor Hanscom's⁷ background is in the field of communications processor design at Sperry Univac, where he worked from 1970 to 1982. Since then he has been responsible for administering the undergraduate Computer Science and Computer Engineering programs at the University of Utah. In that capacity he teaches core computer science classes, serves as faculty advisor for individual students and for the Student Chapter of the Association for Computing Machinery, participates in student recruiting activities, and serves as director of the High School Computing Institute. His interests are in the areas of undergraduate education, computer architecture, and data communications.

- Hanscom, D.H., *Instructor's Manual to Accompany Structures and Abstractions, An Introduction to Computer Science with Pascal*, by William I. Salmon, Richard D. Irwin, 1991, 393 pages.

Charles Hansen

Associate Professor, School of Computing
Ph.D., University of Utah, 1987

⁶<http://www.cs.utah.edu/~ganesh/>

⁷<http://www.cs.utah.edu/~hanscom/>

Professor Hansen⁸ joined the faculty in 1998. His research interests span scientific visualization, computer graphics, and high performance computing. Scientific visualization of large scale problems is of key interest and recent work involves taking into consideration time-varying data and exploiting this for speeding up the visualization process. Other methods for visualizing large amounts of data are multiresolution models and view dependent algorithms. The interest in computer graphics is driven from the scientific visualization perspective but includes parallel algorithms for speeding up global illumination.

- J. Kniss, P. McCormick, A. McPherson, J. Ahrens, J. Painter, A. Keahey, and C. Hansen. TRex, Texture-based Volume Rendering for Extremely Large Datasets. In *IEEE Computer Graphics and Applications*, July 2001.
- J. Kniss, G. Kindlmann, and C. Hansen. Interactive Volume Rendering Using Multi-Dimensional Transfer Functions and Direct Manipulation Widgets. in *IEEE Visualization 2001*, 2001.
- P. Sutton and C. Hansen. Accelerated Isosurface Extraction in Time-varying Fields. in *IEEE Transactions on Visualization and Computer Graphics*, 2000.
- S. Parker, M. Parker, Y. Livnat, P-P. Sloan, C. Hansen, and P. Shirley. Interactive Ray Tracing for Volume Visualization. In *IEEE Transactions on Visualization and Computer Graphics*, 1999.
- T. Udeshi and C. Hansen. Towards interactive photorealistic rendering of indoor scenes: A hybrid approach. In *Eurographics Rendering Workshop 1999*.

Thomas C. Henderson

Professor, School of Computing
Ph.D., University of Texas, 1979

Professor Henderson's⁹ professional interests include autonomous agents, multisensor systems, and simulation. Major areas of current research are robot behavior specification, simulation, multisensor integration, and bio-based computational models. Prior to his arrival at Utah, he was a visiting professor at the Institut National de Recherche en Informatique et en Automatique (INRIA), France, and a Research Associate at the Institut fuer Nachrichtentechnik, Deutsche Forschungs und Versuchsanstalt fuer Luft und Raumfahrt (DFVLR), Germany.

- "Smart Sensor Snow," Thomas C. Henderson, Mohamed Dekhil, Scott Morris, Yu Chen and William B. Thompson, IEEE Conference on Intelligent Robots and Intelligent Systems, Victoria, CA, 13-16 Oct, 1998.
- "Instrumented Sensor System Architecture," Mohamed Dekhil and Thomas C. Henderson, *International Journal of Robotics Research*, Vol. 17, No. 4, April, pp. 402-417, 1998.
- "Flat Surface Reconstruction Using Sonar," Thomas C. Henderson, Mohamed Dekhil, Beat Brüderlin, Larry Schenkat and Larkin Veigel, *International Journal of Robotics Research*, Vol 17, No. 5, May, pp. 504-511, 1998.
- "Evolutionary Teleomorphology," *Journal of Robotics and Autonomous Systems*, Vol. 19, No. 1, November 1996, pp. 23-32. (Thomas C. Henderson and Alexei A. Efros).
- Henderson T. C. with S. Susswein, J. Zachary, C. Hansen, P. Hinker, and G. Marsden, "Parallel Path Consistency," *International Journal of Parallel Programming*, Vol. 20, No. 6, 1992.

Lee A. Hollaar

Professor, School of Computing
Ph.D., University of Illinois at Urbana-Champaign, 1975

Professor Hollaar's¹⁰ primary interest is in legal issues regarding computers, particularly the intellectual property protection of software and information. As a Fellow with the Committee on the Judiciary, he has advised the United States Senate on

⁸<http://www.cs.utah.edu/~hansen/>

⁹<http://www.cs.utah.edu/~tch/>

¹⁰<http://www.cs.utah.edu/~hollaar/>

computer-related issues such as encryption, copyright and patent, and regulation of the internet. He was one of the drafters of the Utah Digital Signature Act, the first law in the world to legally recognize digital signatures, and is active in the implementation of the required infrastructure. He directed the Utah Retrieval System Architecture (URSA) project, which developed hardware and software systems to support large information retrieval systems, including a special-purpose VLSI processor for the rapid searching of text and one of the first workstation-based client-server distributed systems for information retrieval. He was also the University's director of campus networking, and continues to work in communications networks and distributed systems.

- Hollaar, L.A., *Now That the CDA's History, Let's Plan Anew*, The National Law Journal, July 14, 1997.
- Hollaar, L.A., *Justice Douglas Was Right: The Need for Congressional Action on Software Patents*, AIPLA Quarterly Journal, Winter 1996.
- Hollaar, L.A., *Method for Error Recovery in a Digital Data Communications System*, United States Patent 5,396,613, March 7, 1995.
- Hollaar, L.A., *Special-Purpose Hardware for Information Retrieval*, in *Information Retrieval, Data Structures and Algorithms*, Prentice-Hall, 1992.
- Hollaar, L.A., *Direct Implementation of Asynchronous Control Units*, IEEE Trans. on Computers, Dec. 1982.

John M. Hollerbach

Professor, School of Computing
Ph.D., Massachusetts Institute of Technology, 1978

Professor Hollerbach's¹¹ interests include robotics and virtual reality. The focus in virtual reality is on improving the transparency and sense of immersion through better mechanical interfaces and their control, better visual and auditory displays, and sensorimotor integration. Haptic interfaces are being employed for virtual manipulation of mechanical CAD models and for scientific visualization. The Treadport locomotion interface is being employed for walk-through synthetic environments such as outdoor terrain.

- Christensen, R., Hollerbach, J.M., Xu, Y., and Meek, S., "Inertial force feedback for the Treadport locomotion interface," *Presence: Teleoperators and Virtual Environments*, 9, 2000, pp. 1-14.
- Hollerbach, J.M., "Some current issues in haptics research," *Proc. IEEE Intl. Conf. Robotics and Automation*, San Francisco, April 24-28, 2000, pp. 757-762.
- Hollerbach, J.M., "Locomotion interfaces," in: *Handbook of Virtual Environments Technology*, K.M. Stanney, ed., Lawrence Erlbaum Associates, Inc., 2002, pp. 239-254.
- Hollerbach, J.M., Thompson, W.B., and Shirley, P., "The convergence of robotics, vision, and computer graphics for user information," *Intl. J. Robotics Research*, 18, 1999, pp. 1088-1100.
- Mills, R., Hollerbach, J.M., and Thompson, W.B., "The biomechanical fidelity of slope simulation on the Sarcos Treadport using whole-body force feedback," *Experimental Robotics VII*, D. Rus and S. Singh, eds., Springer-Verlag London, 2001, in press.

Wilson C. Hsieh

Assistant Professor, School of Computing
Ph.D., Massachusetts Institute of Technology, 1995

Professor Hsieh¹² joined the faculty in September 1997. His research interests are in compilers and programming languages, operating systems, and architecture.

¹¹<http://www.cs.utah.edu/~jmh/>

¹²<http://www.cs.utah.edu/~wilson/>

- Sean McDirmid, Matthew Flatt, Wilson C. Hsieh, “Jiazzi: New-Age Components for Old-Fashioned Java.” In *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications 2001*, October 2001.
- Wilson C. Hsieh, Dawson R. Engler, Godmar Back, “Reverse-Engineering Instruction Encodings.” In *Proceedings of the 2001 USENIX Technical Conference*, June 2001.
- Godmar Back, **Wilson C. Hsieh**, Jay Lepreau, “Processes in KaffeOS: Isolation, Resource Management, and Sharing in Java.” In *Proceedings of the Fourth Symposium on Operating System Design and Implementation*, October 2000.
- John B. Carter, Zhen Fang, Wilson C. Hsieh, Sally A. McKee, and Lixin Zhang, “The Impulse Memory System.” To appear in *IEEE Transactions on Computers*, Special Issue on High Performance Memory Systems.
- Massimiliano Poletto, Wilson C. Hsieh, Dawson R. Engler, and M. Frans Kaashoek, “ ‘C and tcc: A Language and Compiler for Dynamic Code Generation.” *ACM Transactions on Programming Languages and Systems*, 21(2), March 1999.

Christopher R. Johnson

Professor, School of Computing
 Research Associate Professor of Physics and Bioengineering
 Ph.D., University of Utah, 1989

Professor Johnson's¹³ research interests are in the area of scientific computing. Particular interests include inverse and imaging problems, adaptive methods for partial differential equations, numerical analysis, problem solving environments, computational problems in medicine, and scientific visualization. In 1992, Professor Johnson was awarded a Young Investigator's Award from the NIH, in 1994 he was awarded the National Young Investigator (NYI) Award from the NSF, and in 1995 he was awarded the Presidential Faculty Fellow (PFF) Award from the NSF. In 1996 he received a DOE Computational Science Award and in 1997 received the Par Excellence Award from the University of Utah Alumni Association and the Presidential Teaching Scholar Award. In 1999, Professor Johnson was Awarded the Governor's Medal for Science and Technology from Utah Governor Michael Leavitt. He directs the Scientific Computing and Imaging Institute.

- R. Westermann, C.R. Johnson, and T. Ertl. Topology Preserving Smoothing of Vector Fields. *IEEE Transactions on Visualization and Computer Graphics*, 2001 (to appear).
- C.R. Johnson, M. Mohr, U. Ruede, A. Samsonov and K. Zyp. Multilevel methods for inverse bioelectric field problems. Yosemite Workshop on Multilevel Methods, Springer-Verlag Lecture Notes in Computer Science, 2001 (to appear).
- J.D. Brederson, M. Ikits, C. Johnson, and C. Hansen. A Prototype System For Synergistic Data Display. In *IEEE Virtual Reality 2001*, Special Topics Workshop, The Future of VR and AR Interfaces: Multi-modal, Humanoid, Adaptive and Intelligent, 2001 (to appear).
- C.R. Johnson, Y. Livnat, L. Zhukov, D. Hart, and G. Kindlmann. Computational Field Visualization. In *Mathematics Unlimited: 2001 and Beyond*, B. Engquist and W. Schmid, Editors, Springer-Verlag, pp. 605-630, 2001.
- J. McCorquodale, D. de St. Germain, S. Parker, and C.R. Johnson. The Utah Parallelism Infrastructure: A Performance Evaluation, *High Performance Computing 2001*, pp. 92-97, Seattle, March 2001.

Robert Kessler

Professor, School of Computing
 Ph.D., University of Utah, 1981

Professor Kessler's¹⁴ current research interests are in agents, software engineering, distributed systems, and visual programming. In the early 90's, he founded the Center for Software Science, a state of Utah Center of Excellence. He has also founded several startup companies and is currently involved with an Internet startup company, emWare, as a member of the board. He has served as member-at-large of ACM SIGPLAN and Vice Chairman for Conferences for SIGPLAN. He recently completed a seven year assignment as co-editor-in-chief of the International Journal of Lisp and Symbolic Computation.

- L. Williams and R. Kessler "All I Really Need to Know about Pair Programming I Learned in Kindergarten," accepted with revisions, to appear in Communications of the ACM.
- N. Dykman, M. Griss, and R. Kessler "Nine Suggestions for Extending UML Extensibility," to appear in UML 99 (October 1999).
- M. Griss, G. Bolcher, Q. Chen, R. Kessler, and L. Osterweil "Agents and Workflow – An Intimate Connection, or Just Friends?" to appear Tools 99 (August 99), Santa Barbara, CA.
- T. Henderson, M. Dekhil, R. Kessler, and M. Griss "Sensor Fusion," in "Control Problems in Robotics and Automation," Edited by Bruno Siciliano and Kimon Valavanis, Springer-Verlag, London, Volume in the Series of Lecture Notes in Control and Information Science, Series Editor: Prof. M. Thoma, 1998, pp. 193-207.

¹³<http://www.cs.utah.edu/~crj/>

¹⁴<http://www.cs.utah.edu/~kessler/>

- R. Kessler and M. Griss “Building Object-Oriented Instrument Kits,” Object Magazine, April 1996. [Also available as HP Laboratories Technical Report - HPL-96-22.]

Mike Kirby

Assistant Professor, School of Computing
Ph.D., Brown University, 2002

Professor Kirby¹⁵ joined the faculty in 2002. His research focus is on large-scale scientific computation and visualization, with an emphasis on the scientific cycle of mathematical modeling, computation, visualization, evaluation, and understanding. His primary research interests include computational science and engineering, high-order numerical methods, concurrent programming, scientific visualization, and high performance computing.

Arthur H. Lee

Clinical Associate Professor, School of Computing
Ph.D., University of Utah, 1992

Professor Lee¹⁶ joined the faculty in 2001. His interests revolve around object-oriented programming. Of particular interest are software design, software architecture, and software evolution. These are being experimented in areas such as programming languages, database systems, and distributed systems. He is also interested in improving computer science curricula. He tries to bring emerging ideas from research and industry into the classroom, always looking for new ones to incorporate into the CS curricula. Some of the emerging ideas that Professor Lee is investigating include aspect-oriented programming, web programming, and embedded programming. He is also interested in finding innovative and effective ways to use computers and other technologies in teaching.

- J.G. Park and A.H. Lee. Specializing The Java Object Serialization Using Partial Evaluation for Faster RMI. In *International Conference on Parallel and Distributed Systems*. IEEE Computer Society Press, June 2001.
- A.L. Lee and H.Y. Shin. Building a Persistent Object Store using the Java Reflection API. In *OOPSLA Workshop on Reflective Programming in C++ and Java*, October 1998.
- A.H. Lee and J.L. Zachary. Adding Support for Persistence to CLOS via its Metaobject Protocol. *Lisp and Symbolic Computation: An International Journal*, June 1997.
- C. Maeda, A.H. Lee, G. Murphy, and G. Kiczales. Open Implementation Analysis and Design. In *Symposium on Software Reusability*. ACM Press, May 1997.
- A.H. Lee and J.L. Zachary. Reflections on Metaprogramming. *IEEE Transactions on Software Engineering*, November 1995.

Jay Lepreau

Research Associate Professor, School of Computing
B.S., University of Utah, 1983

Professor Lepreau’s¹⁷ research interests focus on operating systems, but expand into many other areas including security, networking, component software, programming and domain-specific languages, compilers, distributed systems, and software assurance. As head of the Flux Research Group, he currently leads three DARPA and NSF-sponsored research projects. The “Alchemy” project is developing a new model of component programming for embedded and other low-level systems. Utah’s Active Networks effort is attempting to develop a router OS that can safely and speedily “execute” Java bytecode-carrying packets. Finally, in a related effort his group is constructing a unique research instrument: a remotely configurable 1000-node network testbed and emulation facility. In these efforts, his group has developed much software, including the “Janos” active network OS, the “Knit” component composition language, the OSKit, the Flick IDL compiler, the Fluke/Flask OS, and the “Alta” and “KaffeOS” Java operating systems. In 1994 he founded the prestigious *Usenix/ACM/IEEE Symposium on Operating Systems Design and Implementation* (OSDI) conference series, and served as its first program chair.

¹⁵<http://www.cs.utah.edu/~kirby/>

¹⁶<http://www.cs.utah.edu/~alee/>

¹⁷<http://www.cs.utah.edu/~lepreau/>

- Alastair Reid, Matthew Flatt, Leigh Stoller, Jay Lepreau, Eric Eide. “Knit: Component Composition for Systems Software.” In *Proc. of the Fourth Symposium on Operating Systems Design and Implementation*, October 2000.
- Godmar Back, Wilson Hsieh, Jay Lepreau. “Processes in KaffeOS: Isolation, Resource Management, and Sharing in Java.” In *Proc. of the Fourth Symposium on Operating Systems Design and Implementation*, October 2000.
- Ray Spencer, Stephen Smalley, Peter Loscocco, Mike Hibler, Dave Andersen, Jay Lepreau. “The Flask Security Architecture: System Support for Diverse Security Policies.” In *Proc. of the Eighth Usenix Security Symposium*, August 1999.
- Bryan Ford, Mike Hibler, Roland McGrath, Patrick Tullmann, Jay Lepreau. “Interface and Execution Models in the Fluke Kernel.” In *Proc. of the Third Symposium on Operating Systems Design and Implementation*, February 1999.
- Bryan Ford, Godmar Back, Greg Benson, Jay Lepreau, Albert Lin, Olin Shivers. “The Flux OSKit: A Substrate for OS and Language Research.” In *Proc. of the 16th ACM Symposium on Operating Systems Principles*, October 1997.
- Eric Eide, Kevin Frei, Bryan Ford, Jay Lepreau, Gary Lindstrom. “Flick: A Flexible, Optimizing IDL Compiler.” In *Proc. of the ACM SIGPLAN 1997 Conference on Programming Language Design and Implementation*, June 1997.

Gary E. Lindstrom

Professor, School of Computing
Ph.D., Carnegie-Mellon University, 1971

Professor Lindstrom’s¹⁸ research interests include programming languages, databases, and scientific data management. He is on the editorial board of *International Journal of Parallel Programming*, and was Editor-in-Chief from its founding until 1993. With Doug DeGroot, he co-edited the book *Logic Programming: Functions, Relations and Equations* published by Prentice-Hall. Professor Lindstrom has been a member of the National Science Foundation Computer and Computation Research Advisory Committee, and served as a Distinguished Visitor of the IEEE Computer Society. In 1981 he received the College of Engineering Outstanding Teaching Award.

- Angela Violi, Xiaodong Chen, Gary Lindstrom, Eric Eddings, Adel F. Sarofim, “Validation Web Site: a Combustion Collaboratory over the Internet”, International Conference on Computational Science (ICCS 2001), San Francisco, May 28-30, 2001, Springer Lecture Notes in Computer Science, V.N. Alexandrov et al. eds.
- Anand Ranganathan, Yury Izrailevsky, Sai Susarla, John Carter, Gary Lindstrom, “Supporting Persistent C++ Objects in a Distributed Storage System”, Workshop on Compiler Support for System Software (WCSS ’99), Atlanta, May 1999.
- Jon Oler, Gary Lindstrom, and Terence Critchlow. Migrating relational data to an OODB: strategies and lessons from a molecular biology experience. In *Proc. of the Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Atlanta, GA, October 1997. ACM SIGPLAN.
- Eric Eide, Kevin Frei, Bryan Ford, Jay Lepreau, and Gary Lindstrom. Flick: a flexible, optimizing IDL compiler. In *Proc. of the Symposium on Programming Language Design and Implementation*, pages 44–56, Las Vegas, NV, June 1997. ACM SIGPLAN.
- Guruduth Banavar and Gary Lindstrom. An application framework for modular composition tools. In *Proc. of the European Conference on Object-Oriented Programming*, pages 91–113, Linz, Austria, July 1996. Springer LNCS 1098.

Emil Praun

Assistant Professor, School of Computing
Ph.D., Princeton University, 2002

¹⁸<http://www.cs.utah.edu/~gary/>

Professor Praun¹⁹ joined the faculty in 2002. His main interests are in computer graphics. In particular, he has been working on applications of local parameterizations of triangle meshes: dividing 3D-embedded surfaces into collections of patches and “unfolding” each of the patches into the plane. He is also interested in combining direct acquisition of 3D surfaces with procedural detail generation methods. He is a recipient of the John E. and Marva M. Warnock Presidential Endowed Chair for Faculty Innovation in Computer Science.

- E. Praun, W. Sweldens, and P. Schröder, “Consistent mesh parameterizations”, *Proceedings of SIGGRAPH 2001*, pages 179-184.
- E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, “Real-time hatching”, *Proceedings of SIGGRAPH 2001*, pages 417-424.
- E. Praun, A. Finkelstein, and H. Hoppe, “Lapped textures”, *Proceedings of SIGGRAPH 2000*, pages 465-470.
- E. Praun, H. Hoppe, and A. Finkelstein, “Robust mesh watermarking”, *Proceedings of SIGGRAPH 1999*, pages 49-56.

Richard F. Riesenfeld

Professor, School of Computing
Ph.D., Syracuse University, 1973

Professor Riesenfeld²⁰ has been involved in research in the areas of computer graphics, animation, computer aided geometric design and CAD/CAM since joining the faculty in 1972. Recently he has been investigating a broad spectrum of research problems in computer graphics, geometric modeling, and manufacturing within an integrated experimental testbed system motivated by the unifying principles of spline theory.

- R.F. Riesenfeld, R. Fish, and S. Drake, “A Case Study in Multi-disciplinary Distributed Collaborative Design,” in Proc. ASME Conference on Network-Centric CAD, in press, Sept. 1997.
- M. Sturgill, E. Cohen, and R. Riesenfeld, “Feature Based 3-D Sketching for Early Design,” Proceedings of the 1995 ASME Computers in Engineering Conference
- R. Riesenfeld, “Some Video Related High Bandwidth Communications Needs,” in Proc. NSF Workshop on vBNS and the Research Agenda for Networking and Applications, June 1995.
Fundamental Developments of Computer Aided Geometric Design, L. Piegl (ed.), Academic Press, 1993.
- Y.C. Hsieh, R. F. Riesenfeld, and S. H. Drake, “Reconstruction of Sculptured Surfaces using Coordinate Measuring Machine”, ASME Design Automation Conference, 1993.

Ellen M. Riloff

Associate Professor, School of Computing
Ph.D., University of Massachusetts at Amherst, 1994

Professor Riloff²¹ joined the faculty in 1994. Her research interests are in natural language processing, machine learning, and artificial intelligence. She is particularly interested in techniques for automatically generating dictionaries and knowledge bases for natural language processing. Most of her work revolves around the task of *information extraction*, which involves extracting information from text. The NLP research group at Utah has built its own NLP system called Sundance, which is a partial parser that activates and instantiates case frames for information extraction. Current research projects include bootstrapping techniques for learning extraction patterns, corpus-based techniques for learning semantic dictionaries, and corpus-based methods for coreference resolution.

- Riloff, E. and Jones, R., “Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping,” in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*. 1999.

¹⁹<http://www.cs.utah.edu/~emilp/>

²⁰<http://www.cs.utah.edu/~rfr/>

²¹<http://www.cs.utah.edu/~riloff/>

- Bean, D. and Riloff, E., “Corpus-Based Identification of Non-Referential Noun Phrases,” in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*. 1999.
- Riloff, E. “Information Extraction as a Stepping Stone toward Story Understanding,” in *Computational Models of Reading and Understanding*. The MIT Press. 1999.
- Riloff, E. and Shepherd, J., “A Corpus-Based Approach for Building Semantic Lexicons,” in *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. 1997.
- Riloff, E., “Automatically Generating Extraction Patterns from Untagged Text,” in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*. 1996.

Peter Shirley

Associate Professor, School of Computing
Ph.D., University of Illinois, 1991

Professor Shirley²² joined the faculty in 1996. He is interested in creating highly realistic images of virtual environments, and visualization of complex data. The former involves explicit and procedural generation of geometric models with realistic optical characteristics, light transport simulation to determine the outgoing light distribution from surfaces, and tone reproduction to create images displayable on low dynamic range media such as paper and CRTs. The latter involves issues of visual representation of complex data, as well as strategies for navigation and interaction that help the user extract local and global information about the data. In 2000 Professor Shirley received the University of Utah Student’s Choice Teaching Award and in 1998 he received the College of Engineering Outstanding Teacher Award.

- H. Jensen, M. Stark, S. Premoze, P. Shirley, F. Durand, and J. Dorsey. A Physically-Based Night Sky Model. In *ACM SIGGRAPH Annual Conference*, 2001.
- M. Ashikhmin, S. Premoze, and P. Shirley. A Microfacet-based BRDF Generator. In *ACM SIGGRAPH Annual Conference*, 2000.
- A. J. Preetham, P. Shirley, and B. Smits. A Practical Analytic Model for Daylight. In *ACM SIGGRAPH Annual Conference*, 1999.
- A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A Non-photorealistic Lighting Model for Automatic Technical Illustration. In *ACM SIGGRAPH Annual Conference*, 1998.
- J. Ferwerda, S. Pattanaik, P. Shirley, and D. Greenberg. A Model of Visual Masking for Computer Graphics. In *ACM SIGGRAPH Annual Conference*, 1997.

Kris Sikorski

Professor, School of Computing
Ph.D., University of Utah, 1982

Professor Sikorski’s²³ current research interests are in the areas of distributed parallel scientific computation and computational complexity with emphasis on information based complexity. Of specific interest are applied problems in geophysics (3-D modeling of earthquakes), combustion (fluid mechanics), and electromagnetic wave propagation (Maxwell equations). Various parallel explicit and implicit algorithms are being studied and implemented on massively parallel machines. Information based complexity is a study of optimal algorithms for problems which are approximately solved, because of partial and contaminated information. Optimal algorithms for solving nonlinear problems with use of various error criteria are of special interest to Professor Sikorski.

- Sikorski, K., Tsay, C., and Wozniakowski, H., *An Ellipsoid Algorithm for the Computation of Fixed Points*, Journal of Complexity, March, 1993.

²²<http://www.cs.utah.edu/~shirley/>

²³<http://www.cs.utah.edu/~sikorski/>

- Sikorski, K., Schuster, J., and Tsay, C., *3-Dimensional Modeling of Acoustic and Elastic Wave Propagation on Parallel Computers*, Transputer Research and Applications 1, IOS Press, 1990.
- Sikorski, K., and Wozniakowski, H., *Complexity of Fixed Points I*, Journal of Complexity, December, 1987.
- Boulton, T., and Sikorski, K., *Complexity of Computing Topological Degree of Lipschitz Functions in N-Dimensions*, Journal of Complexity, Vol. 2, pp. 44-59, 1986.
- Sikorski, K., and Wozniakowski, H., *For Which Error Criteria Can We Solve Nonlinear Equations*, Journal of Complexity, 1986.

Konrad Slind

Assistant Professor, School of Computing
Ph.D., TU Munich, 1999

Professor Slind²⁴ joined the faculty in 2001. His research interests are in logic and functional programming. He is particularly interested in higher order logic, its implementation, and its application to deductive verification of system properties. Recent research has investigated the modeling of generic and functional programming. Before coming to Utah, he participated in a European project that developed middleware for applications that require a theorem proving component.

- Konrad Slind. Another Look at Nested Recursion. In *Proceedings of TPHOLs 2000, Springer-Verlag Lecture Notes in Computer Science*, No. 1869.
- Richard Boulton and Konrad Slind. Automatic Derivation and Application of Induction Schemes for Mutually Recursive Functions. In *Proceedings of CL 2000, Springer-Verlag Lecture Notes in Artificial Intelligence*, No. 1861.
- Konrad Slind, Wellfounded Schematic Definitions. In *Proceedings of CADE-17, Springer-Verlag Lecture Notes in Computer Science*, No. 1831.
- Louise Dennis, Graham Collins, Michael Norrish, Richard Boulton, Konrad Slind, Graham Robinson, Mike Gordon, and Tom Melham. The PROSPER Toolkit. In *Proceedings of TACAS 2000. Springer-Verlag Lecture Notes in Computer Science*, No. 1785. Awarded a best paper prize.
- Konrad Slind and Richard Boulton. Iterative Dialogues and Automated Proof. In *Frontiers of Combining Systems 2*, No. 7, D. Gabbay and M. de Rijke (eds). In *Studies in Logic and Computation*, Research Studies Press, 2000.

Frank Stenger

Professor, School of Computing
Ph.D., University of Alberta, 1965

Professor Stenger's²⁵ research interests include the development of new methods of computation and the computer solution of computationally difficult problems from science and engineering, such as inverse problems, crack problems, flow problems and heat problems. He developed the Sinc methods, which provide optimal algorithms in all areas of engineering computation. He is currently writing, jointly with Michael O'Reilly and Tao Zhang, a *Sinc Tool Box* computer-based tutorial package to make these methods accessible to users. One of his students (Kenneth Parker) has recently completed his Ph.D. dissertation *PTOLEMY: A Sinc-Collocation Mapping Sub-System*, which is a computer sub-package of *Maple* that automates the solution of partial differential equations. Several of his students have recently written or are presently writing computer packages for solving a broad range of difficult engineering problem—such as Navier-Stokes equations (Barkey and Vakili, Narasimhan), Maxwell equations (Naghsh-Nilchi) based on his recently discovered Sinc method of approximating indefinite convolutions, which leads to a unified approach for solving elliptic, parabolic, and hyperbolic differential equations. Another student (Ross Schmittlein) is writing a package based on Sinc, constructing conformal maps. Stenger and his former student O'Reilly have been developing methods which make it possible to invert the Helmholtz equation without computing the forward solution. During the next few academic years he expects to extend these inversion methods so that they can be applied to rendering, to visualization, to the determination of paths for robots, and to the inversion of heat and electromagnetic problems for medical and geophysical applications. Seven of his papers have been accepted for publications during this past academic year.

²⁴<http://www.cs.utah.edu/~slind/>

²⁵<http://www.cs.utah.edu/~stenger/>

- R. Kress, I.H. Sloan, and F. Stenger, *A Sinc Quadrature Method for the Double-Layer Integral Equation in Planar Domains With Corners*, with R. Kress and I.H. Sloan, to appear in volume honoring P.M. Anselone's 65th birthday.
- F. Stenger, B. Keys, M. O'Reilly, and K. Parker, *ODE-IVP—PACK, via Sinc Indefinite Integration and Newton Iteration*, to appear in "Numerical Algorithms".
- F. Stenger, *A Unified Approach to the Approximate Solution of PDE*, to appear in "Communications in Applied Analysis", 1998.
- F. Stenger, R. Kress, and I.H. Sloan, *Constructing Conformal Maps via Sinc Methods*, prepared for the Cyprus Proceedings on Computational Complex Variables, Oct., 1997.
- A. Naghsh-Nilchi and F. Stenger, *Solution of the Electric Field Integral Equations via Sinc Convolution*, submitted.

Cynthia A. Thompson

Assistant Professor, School of Computing
Ph.D., University of Texas, 1998

Professor Thompson²⁶ joined the faculty in 2000. Her research interests are in machine learning, natural language processing, and artificial intelligence. She is especially interested in applying machine learning to complex, relational tasks such as language processing and adaptive interfaces. After receiving her Ph.D., she spent two years at the Center for the Study of Language and Information at Stanford University, building and studying conversational interfaces that adapt themselves over the course of several interactions with a user.

- Mehmet Goker and Cynthia Thompson, "The Adaptive Place Advisor: A Conversational Recommendation System," *Proceedings of the 8th German Workshop on Case Based Reasoning*, Lammerbuckel, Germany, March 2000.
- Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney, "Active Learning for Natural Language Parsing and Information Extraction," *Proceedings of the Sixteenth International Machine Learning Conference*, Bled, Slovenia, June 1999. (ICML-99)
- Cynthia A. Thompson and Raymond J. Mooney, "Automatic Construction of Semantic Lexicons for Learning Natural Language Interfaces," *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Orlando, FL, July, 1999. (AAAI-99)
- Cynthia A. Thompson and Raymond J. Mooney, "Inductive Learning For Abductive Diagnosis," *Proceedings of the Twelfth National Conference on AI*, Seattle, WA, July 1994. (AAAI-94)

William B. Thompson

Professor, School of Computing
Ph.D., University of Southern California, 1975

Professor Thompson's²⁷ primary research interest is in the area of visual perception. Currently, he is exploring how an understanding of computational vision can aid in improving the spatial information conveyed by graphical displays. He is also active in the exploration of techniques for analyzing visual motion, sensing for manufacturing, and vision-based navigation. Professor Thompson joined the faculty in 1991 after 16 years in the Computer Science Department at the University of Minnesota.

- C. Madison, W.B. Thompson, D.J. Kersten, P. Shirley, and B.S. Smits, "Use of Interreflection and Shadow for Surface Contact," *Perception and Psychophysics*, 63(2), February, 2001.
- W.B. Thompson, C.M. Valiquette, B.H. Bennett, and K.T. Sutherland, "Geometric Reasoning for Map-Based Localization," *Spatial Cognition and Computation*, 1(3), 1999.

²⁶<http://www.cs.utah.edu/~cindi/>

²⁷<http://www.cs.utah.edu/~thompson/>

- S. Premoze, W.B. Thompson, and P. Shirley, "Geospecific Rendering of Alpine Terrain," *Proceedings of the Eurographics Rendering Workshop*, June 1999.
W.B. Thompson, J.C. Owen, H.J. de St. Germain, Stevan R. Stark, and T.C. Henderson, "Feature-Based Reverse Engineering of Mechanical Parts," *IEEE Transactions on Robotics and Automation*, 15(1), February 1999.
- W.B. Thompson, "Exploiting Discontinuities in Optical Flow," *International Journal of Computer Vision*, 30(3), 1998.

Ross Whitaker

Assistant Professor, School of Computing
Ph.D., University of North Carolina, 1993

Professor Whitaker²⁸ joined the faculty in 2000. He has interests in computer vision, visualization, and image processing. In the area of medical image processing, Dr. Whitaker is a codeveloper of the "Insight" toolkit for segmenting and visualization the 3D color data associated with the Visible Human Project. Dr. Whitaker is also working on new, statistics-based methods for building surface models from noisy range measurements, such as those from laser radar and ultrasound. In the area of visualization, Dr. Whitaker is developing new methods for visualizing biological volume datasets and for processing the surface models that are derived from these datasets.

- R. T. Whitaker, "A Level-Set Approach to Image Blending," *IEEE Transactions on Image Processing*, in press.
- A. Mangan, R. T. Whitaker, "Partitioning 3D surface meshes using watershed segmentation," *IEEE Transactions on Visualization and Computer Graphics*, 5(4), pp. 308–321.
- R. Whitaker, J. Gregor, and P. Chen, "Indoor scene reconstruction from sets of noisy range images," *Second International Conference on 3-D Digital Imaging and Modeling*, October 1999, pp. 348–357.
- R. Whitaker, "A level-set approach to 3D reconstruction from range data," *International Journal of Computer Vision*, 29(3), October 1998, pp. 203–231.
- M. Tuceryan, D. Greer, R. Whitaker, D. Breen, C. Crampton, E. Rose, K. Ahlers, "Calibration requirements and procedures for a monitor-based augmented reality system," *IEEE Transactions on Visualization and Computer Graphics*, 1(3) September 1995, pp. 255–273.

Joseph L. Zachary

Clinical Professor, School of Computing
Ph.D., Massachusetts Institute of Technology, 1987

Professor Zachary²⁹ joined the faculty in 1987. He is interested in finding ways to use computers in teaching science and engineering in general, and computer science in particular. He is currently developing Web-based tools and curricula for teaching introductory programming, computer science, and scientific computing courses. The author of two textbooks in scientific programming, Professor Zachary received the 1999 IEEE Computer Science and Engineering Undergraduate Teaching Award, won the University of Utah Distinguished Teaching Award in 1997, was named a Department of Energy Undergraduate Computational Science Education Award winner in 1996, was a University of Utah Presidential Teaching Scholar in 1995, and received the College of Engineering Outstanding Teaching Award in 1990.

- J. Turner and J.L. Zachary. Javiva: Java Visualization and Validation. In *Papers of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education*. ACM Press, February 2001.
- E. Odekirk and J. Zachary. Automated Feedback on Programs Means Students Need Less Help From Teachers. In *Papers of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education*. ACM Press, February 2001.

²⁸<http://www.cs.utah.edu/~whitaker/>

²⁹<http://www.cs.utah.edu/~zachary/>

- D. Price, E. Riloff, J. Zachary, and B. Harvey. NaturalJava: A Natural Language Interface for Programming in Java. In *Proceedings of the International Conference on Intelligent User Interfaces*. ACM Press, January 2000.
- J.L. Zachary. *Introduction to Scientific Programming: Computational Problem Solving Using Mathematica and C*. TELOS/Springer-Verlag, 1998.
- J.L. Zachary. *Introduction to Scientific Programming: Computational Problem Solving Using Maple and C*. TELOS/Springer-Verlag, 1996.

6

Current Funded Research

As an indication of the breadth and specialties of current research activity in the School of Computing, the following list summarizes a partial list of funded projects underway at the time of this handbook's publication.

- *Application Driven Advancement of Asynchronous Design Methods*, NSF, Brunvand and Gopalakrishnan.
- *Adaptive Structure Aware Memory Systems*, DARPA, Carter.
- *Representation for SSF Multilateral Component Fabrication*, SRI, Cohen.
- *Center for the Simulation of Accidental Fires and Explosions C-Safe*, DOE, Davis, Hansen, Henderson, Johnson, and Lindstrom.
- *Test Model Checking Based Verification of Multiprocessor Cache Protocols*, NSF, Gopalakrishnan.
- *Formal Verification and Design of System-Level Hardware*, NSF, Gopalakrishnan.
- *Formalization and Verification of Computer System Interconnect Busses*, NSF, Gopalakrishnan.
- *Microengine for Programmable Self-Timed Control*, NSF, Gopalakrishnan and Brunvand.
- *MRI Acquisition of an Experimental Testbed for Computer Graphics*, NSF, Hansen, Cohen, Johnson, Shirley, and W. Thompson.
- *Interactive Ray Tracing for Visualization*, NSF, Hansen, Smits, Shirley, and W. Thompson.
- *Undergraduate Experience in Research*, Sandia, Hansen.
- *Rapid Extraction of Isosurfaces from Time-Varying Data*, LLNL, Hansen.
- *CISE Education Innovation: Simulation Science and Education*, NSF, Henderson and Zachary.
- *Virtual Prototyping for Human Centric Design*, NSF, Hollerbach, Cohen, and W. Thompson.
- *CAREER: Compiler Optimizations for Instruction Bandwidth*, NSF, Hsieh.
- *Corridor One - An Integrated Distance Visualization Environment for SSI and ASCII Applications*, DOE, Johnson and Hansen.

- *Bioelectric Field Modeling Stimulation and Visualization*, NIH, Johnson, Hansen, and MacLeod.
- *Interactive Flow Visualization Using Haptics*, NSF, Johnson, Hansen, and Hollerbach.
- *Presidential Faculty Fellowship*, NSF, Johnson.
- *Program for Partnerships for Advanced Computational Infrastructure*, NSF, Johnson.
- *ASCI Advanced Visualization Technology Center*, Argonne Lab, Johnson and Hansen.
- *Robust Composition of Embedded Systems*, DARPA, Lepreau.
- *JANOS: A Java-oriented Active Network Operating System*, DARPA, Lepreau.
- *A Large-Scale Network Emulation Facility*, NSF, Lepreau.
- *An Integrated Repository for Biology Research*, DARPA, Lindstrom.
- *POWRE: Understanding and Improving Memory System Performance*, NSF, McKee.
- *Understanding and Improving Memory System Performance Via a New Approach to Cache Analysis and Access Ordering*, NSF, McKee.
- *Acquisition of Equipment for Telecollaboration Telepresence and Design*, NSF, Riesenfeld.
- *Multiphase Integrated Engineering design (MIND)*, Air Force, Riesenfeld and Cohen.
- *Computer Graphics and Scientific Visualization NSF Science and Technology Center*, NSF, Riesenfeld and Cohen.
- *CAREER: Building Conceptual Natural Language Processing Systems for Practical Applications*, NSF, Riloff.
- *User-Directed Hybrid Deterministic and Monte Carlo Parallel Light Transport Algorithms*, NSF, Shirley and Hansen.
- *Realistic Computer Graphics for Natural Scenes*, NSF, Shirley, Smits, and W. Thompson.
- *CISE Research Infrastructure: Modeling Complex Physical and Computational Environments*, NSF, W. Thompson.
- *CISE Research Instrumentation: Realistic Computer Graphics*, NSF, W. Thompson.
- *CAREER: A Statistical Framework for Estimating 3D Manifolds from Range Data*, NSF, Whitaker.
- *Interactive Level-Set Modeling for Visualization of Biological Volume Datasets*, NSF, Whitaker.
- *The Visible Human project Image Processing Tools*, NLM, Whitaker.
- *Multiresolution Visualization Tools for Interactive Analysis of Large-Scale N-Dimensional Data Sets*, NSF, Whitaker.
- *A Statistical Approach to 3D Terrain Reconstruction - Phase II*, ONR, Whitaker.
- *Helping Beginning Programmers View Their Programs Abstractly*, NSF, Zachary.
- *Creating Interactive Web Content - CPSC 1040*, USHE, Zachary.